MASTER EN SCIENCES INFORMATIQUES

Evaluation des performances de TCP dans les réseaux ad hoc

Deglume, David

Award date:
2002

Facultés Universitaires N. D. de la Paix
Namur
Institut d'Informatique

Evaluation des performances de TCP dans les réseaux ad hoc

David DEGLUME

Promoteur: O. Bonaventure

Mémoire présenté pour l'obtention du grade
de Licencié en Informatique

Année académique 2001-2002

Résumé

Les réseaux ad hoc (MANET) sont des réseaux mobiles sans fil où il n'existe aucune infrastucture prédéfinie. Dans de tels réseaux, c'est la couche de routage qui tente d'assurer la cohésion du réseau en fournissant un service aussi transparent que possibe aux protocoles des couches de plus haut niveau. Le but de ce mémoire est d'observer, à travers une série de scénarii, le comportement de TCP dans un environnement de type MANET. Nos simulations permettent d'affirmer que le débit dépend principalement de la longueur de la route suivie pour acheminer les paquets et de la taille des segments TCP. La taille de la fenêtre TCP, quant à elle, peut être corrélée avec le taux de perte de segments. Nous montrons également que la mobilité des nœuds du réseau influence à la fois le débit et le taux de perte.

Abstract

Ad hoc networks (MANET) are mobile wireless networks that do not have any kind of fixed infrastructure. In such networks, the routing layer tries to tie the network together into a seamless entity and provide transparent service to higher layer protocols. The aim of this thesis is to observe, through a set of scenarios, the behaviour of TCP in a MANET environment. Our simulations show that the TCP throughput depends mainly on the length of the path through the ad hoc network and the TCP segment size. The TCP window size can be correlated to the loss rate. We also show that the mobility of network node has an effect on both the TCP throughput and the segment loss rate.

Table des matières

Introduction

A l'heure actuelle, de plus en plus de personnes possèdent des équipements portables tels que des laptops, des téléphones mobiles ou des lecteurs MP3, pour leurs besoins professionnels ou privés. Dans la plupart des cas, ces équipements sont utilisés séparément, c'est-à-dire que les applications sur ces équipements n'interagissent pas entre elles. Imaginons cependant que cela soit rendu possible avec toutes les applications qui en découleraient : partage de fichiers lors de conférence, jeu vidéo en réseau dans la rue ou encore possibilté de relever sa boîte à courrier électronique à partir de son PDA. Ces exemples de réseaux spontanés, créés à la demande sont rendus possibles grâce aux réseaux ad hoc (MANET). De tels réseaux sont conçus pour opérer dans des environnements où aucune infrastructure prédéfinie n'existe. La communication y est typiquement multi-sauts, chaque nœud pouvant retransmettre le trafic.

De nombreuses recherches sont effectuées à propos des MANETs, la plupart étant orientées vers le développement et les performances des protocoles de routage ad hoc. Cependant, peu de travaux sont consacrés à l'interaction de ces protocoles avec des services de transport tel que TCP. Le but de ce mémoire est donc d'observer le comportement de TCP dans un environnement de type MANET à travers un certain nombre de petits scénarii, volontairement simplifiés de façon à pouvoir vérifier les résultats obtenus expérimentalement. Dans ces simulations, l'ensemble du trafic est généré par une seule connexion TCP afin d'éviter le trafic concurrent. Le protocole de routage DSR est utilisé pour acheminer les paquets à l'intérieur du réseau.

L'organisation du mémoire est la suivante : tout d'abord, une présentation des réseaux ad hoc sera entreprise, donnant une définition et les principales caractéristiques de tels réseaux pour ensuite en donner un historique et les principales applications. Le chapitre 2 présente la problématique des protocoles de routage dans un environnement ad hoc. Une présentation de DSR, qui est le protocole choisi pour réaliser nos simulations, vient ensuite conclure le chapitre. Le chapitre 3 traite du service de transport TCP en y présentant les principaux mécanismes et caractéristiques. Le chapitre 4 reprend l'ensemble des résultats obtenus lors des simulations. Enfin, après

la conclusion, une série d'annexes est proposée pour donner un complément d'information sur certains éléments du mémoire qui n'ont pu y être développé dans les chapitres principaux.

Chapitre 1

Les réseaux mobiles ad hoc

1.1 Définition

Le concept de réseau ad hoc est présent dans le monde de la recherche scientifique depuis le début des années 70. Etant donné que ce domaine est depuis lors en perpétuelle évolution, les besoins, les applications mais aussi les attentes attachées à de tels réseaux ne cessent de changer. Dans cette optique, définir le concept de réseau ad hoc n'est donc pas quelque chose de trivial. Alors que [28] isole trois définitions qu'il qualifie de pertinentes, dans le but de les analyser et d'en extraire une définition canonique, on se contentera ici de reprendre la définition proposée par l'Internet Engeneering Task Force (IETF), l'organisme responsable de l'évolution de l'Internet. Cet organisme a d'ailleurs créé un groupe de travail à l'intention des réseaux mobiles ad hoc et les définit de la manière suivante :
«un réseau mobile ad hoc (MANET) est un système autonome de routeurs en mouvement (avec leurs hôtes associés) connectés entre eux par des liens sans fil - l'ensemble formant un graphe arbitraire. Les routeurs sont libres de se mouvoir de la manière dont ils le souhaitent et s'organisent automatiquement de manière arbitraire ; par conséquence, la topographie du réseau sans fil est susceptible de se modifier rapidement et de manière imprévisible. De tels réseaux opèrent de manière autonome, ou peuvent être connectés à l'Internet.» [2]

A la lumière de cette définition, on remarque que la notion de réseau est «redéfinie» à l'aide des mots «système» et «union». De plus, cette définition ne présuppose aucune technologie de communication particulière en utilisant le terme générique «routeur»[1]. Dans ce sens, la définition ci-dessus cristallise donc la nature multi-sauts des réseaux ad hoc. On insistera aussi sur l'absence d'infrastucture prédéfinie comme le suggère la dernière phrase

[1] D'un point de vue technologique, un routeur fait ici référence à l'appareillage, qui dans un réseau, est capable de relayer du trafic.

de la définition. Enfin, soulignons que le concept de mobilité des participants est également présent.

1.2 Historique

L'origine des réseaux ad hoc remonte jusqu'en 1968, avec les premiers travaux initiés dans le cadre du réseau ALOHA ; l'objectif de ce réseau était à l'époque de relier entre eux différents postes de travail dispersés dans l'archipel d'Hawaï. Le protocole ALOHA s'appuyait sur des stations fixes et fonctionnait de manière distribuée en utilisant une méthode d'accès au canal de transmission appropriée au mode de fonctionnement ad hoc. Signalons également qu'il s'agit d'un protocole à saut unique, ce qui veut dire qu'aucun mécanisme de routage n'est nécessaire pour l'acheminement des données, chaque entité du réseau étant directement joignable par n'importe quel autre participant du réseau.

Inspiré par le réseau ALOHA et les premiers développements de réseaux fixes à commutation de paquets, le Defense Advanced Research Project Agency (DARPA) commença, en 1973, à travailler sur le Packet Radio Network (PRNet) qui est un réseau multi-sauts basé sur une architecture distribuée utilisant des techniques de routage de type store-and-forward. L'arrivée du multi-sauts permet maintenant de connecter des stations qui ne sont pas à portée radio l'une de l'autre. De plus, le multi-sauts permet d'augmenter la capacité du réseau en permettant la réutilisation spatiale des ressources pour les connexions concurrentes.

En 1983, apparu le Survivable Radio Network (SURAN) dont les principales motivations étaient les suivantes : d'abord développer des appareillages radio plus petits, meilleurs marchés, consommant moins d'énergie et pouvant supporter des protocoles de routage plus complexes. On souhaite aussi valider un certain nombre d'algorithmes pour la gestion de plusieurs milliers de nœuds dans un réseau et améliorer la sécurité dans les réseaux radio.

En 1987, Low-cost Packet Radio (LPR) élaboré sur base de la dynamique des clusters visait à améliorer la sécurité et augmenter la capacité des réseaux radio.

Bien qu'au fil du temps beaucoup de projets similaires furent développés, tous ces systèmes sans fil furent boudés par les consommateurs. C'est alors qu'en développant le IEEE 802.11, un standard pour les réseaux locaux sans fil (WLAN), l'Institute of Electrical and Electronic Engeneering (IEEE) remplaça l'appelation Packet-Radio Network par celle de Ad Hoc Network. Les Packet-Radio Networks étant jusqu'alors associés aux réseaux

multi-sauts militaires ou utilisés par les services d'urgence. En changeant de nom, l'IEEE espérait ainsi changer l'image de marque de ces réseaux.

A l'heure actuelle, les réseaux ad hoc commencent à s'intégrer à notre quotidien en offrant de plus en plus de possibilités en termes de technologie de communication (voir 1.4. Applications).

1.3 Caractéristiques des MANETs

Les MANETS possèdent plusieurs caractéristiques qui font aujourd'hui l'objet de nombreuses recherches. Les caractéristiques suivantes [8, 10] forment un ensemble de paramètres prédominants pour la conception des protocoles utilisés dans de tels réseaux.

1. **L'absence d'infrastructure**
   Les réseaux ad hoc se distinguent des autres réseaux mobiles par leur absence d'une structure préexistante et de tout genre d'administration centralisée. Ils ne peuvent donc pas s'appuyer sur une infrastructure de base pour réaliser leurs fonctions de base telles que le routage ou la securité. Ce sont les entités mobiles qui sont responsables d'établir et de maintenir la connectivité du réseau d'une manière continue et ce de manière distribuée.
2. **Une topologie dynamique**
   Les nœuds sont libres de se mouvoir arbitrairement, ce qui implique que la topographie (typiquement multi-sauts) peut changer n'importe quand, rapidement et de manière aléatoire.
3. **Les liaisons ont un débit variable et une bande passante limitée**
   Les liaisons sans fil ont intrinsèquement une capacité inférieure à celle de leurs homologues cablés. De plus, le débit utile (c'est-à-dire celui après avoir déduit les effets des accès multiples, du fading, du bruit, des interférences, ...) est souvent largement inférieur au taux de transfert maximum permis par le mode de transmission. Une conséquence de ces faibles débits de liaison est que fréquemment on approchera (ou même parfois dépassera) la capacité maximum du réseau pouvant ainsi mener à un phénomène de congestion. En effet, un réseau mobile sans fil est souvent une simple extension d'un réseau fixe. Or les applications distribuées sont de plus en plus exigeantes en terme de capacité de réseau.
4. **L'utilisation limitée de l'énergie**
   Une partie, voire l'entièreté des nœuds d'un MANET, peut reposer sur l'emploi de batteries ou d'un autre moyen limité pour puiser son énergie. Pour ces nœuds, il faudra donc mettre en œuvre des solutions

pour minimiser la consommation. Cette réserve limitée d'énergie aura pour conséquence que ces nœuds ne pourront prendre part de manière continue à tous les échanges d'information à l'intérieur du réseau sous peine de voir leur autonomie fortement réduite. Les mécanismes utilisés dans de tels réseaux doivent donc être aussi optimisés pour préserver les ressources d'énergie.

Enfin, citons en plus de ces caractéristiques le fait que certains réseaux (par exemple des réseaux militaires) pourront être relativement étendus et denses, c'est-à-dire comporter plusieurs centaines de nœuds par unité de surface. Il existe donc un réel besoin d'adaptabilité par rapport à la taille (scalability).

1.4 Applications

Grâce à leur facilité et leur rapidité de déploiement, les MANETs peuvent couvrir un champ d'application très large, et ce dans un grand nombre de domaines [10, 25]. Ce type de réseau est de manière générale particulièrement indiqué dans les cas où on ne peut (veut) pas déployer ou gérer une infrastructure cablée, parce que cela est trop contraignant ou impossible.

Les applications ont d'abord été considérées par les militaires et les services d'urgence où le fait de posséder un réseau décentralisé et sans structure fixe à déployer représente un avantage opérationnel certain et même parfois une réelle nécessité. Dans le secteur commercial, les équipements sans fil et de communication mobile commencent seulement à devenir financièrement accessibles. Les réseaux ad hoc dans un tel secteur pourraient également être utilisés dans des situations où aucune infrastruture (fixe ou cellulaire) n'est disponible, c'est-à-dire par exemple dans des zones isolées tels que des sites archéologiques. Les réseaux ad hoc pourraient aussi servir à étendre des réseaux publics en zone urbaine qui seraient équipés de points d'accès sans fil, permettant ainsi son déploiement rapide tout en offrant des services variés tels que du conferencing, de l'échange de fichiers ou encore la possibilité de jouer en réseau. Certains de ces points d'accès pourraient également servir de relais entre les différents utilisateurs, d'autres servant de portail vers un backbone d'entreprise par exemple.

A une échelle plus restreinte, les réseaux ad hoc peuvent se former pour relier entre eux une série d'équipements tels que des notebooks ou des palmtops pour former de véritables Personnal Area Networks (PAN) où l'on pourrait ainsi s'échanger de l'information (mise à jour d'une configuration, alarme, données communes pour tous les participants d'une même conférence, ...). Peut-être pouvons-nous même envisager des applications où l'on pourrait, à l'aide de tels réseaux, interconnecter plusieurs robots capables

de s'échanger de l'information et qui s'occuperaient par exemple des tâches ménagères de base mais aussi des tâches de surveillance et de maintenance de la sécurité.

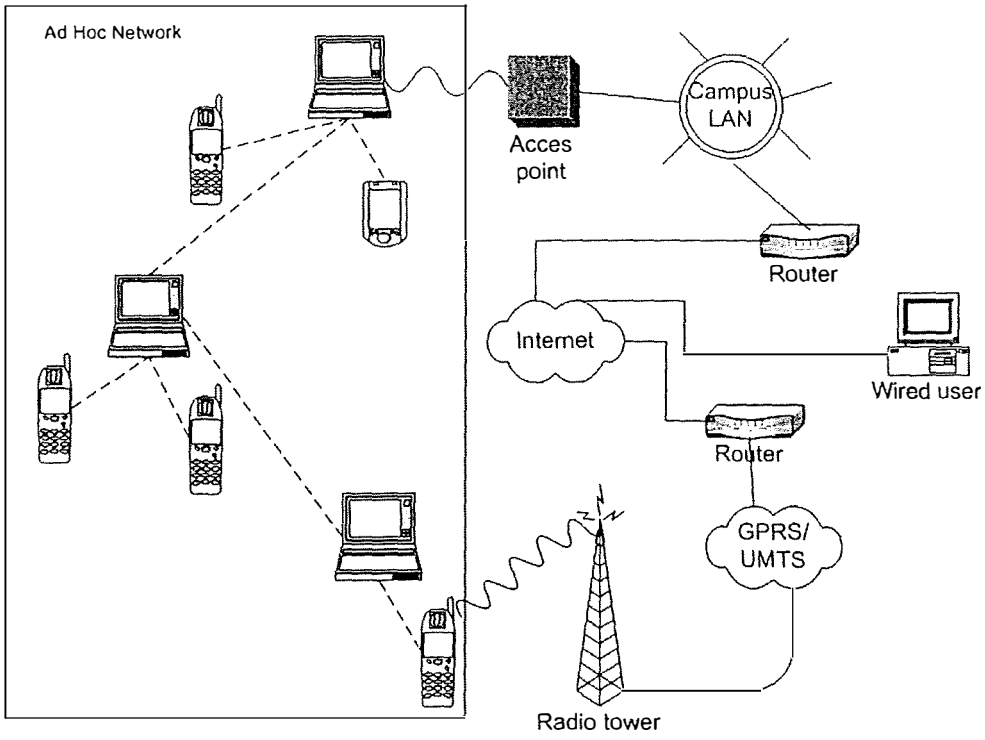


FIG. 1.1 – Exemple de réseau ad hoc formé de trois PAN. Deux de ces PAN possèdent un point d'accès vers l'Internet : l'un grâce à un point d'accès de style WLAN donnant également accès au backbone d'une université, l'autre via un téléphone équipé de la technologie GPRS/UMTS .

Chapitre 2

Les protocoles de routage ad hoc

De manière générale, le routage est une méthode d'acheminement de l'information vers une destination à l'intérieur d'un réseau donné. Le problème du routage consiste pour un réseau à déterminer un chemin optimal pour l'acheminement des paquets à travers celui-ci en fonction de certains critères de performance.

2.1 Le problème du routage dans les réseaux ad hoc

Un réseau ad hoc doit donc s'organiser automatiquement de façon à être déployable rapidement pour pouvoir s'adapter aux conditions de trafic et aux différents mouvements des unités mobiles.

Dans le but d'assurer la connectivité du réseau malgré l'absence d'infrastructure fixe et la mobilité des stations, chaque nœud est susceptible d'être mis à contribution pour participer au routage et pour retransmettre les paquets d'un nœud qui n'est pas en mesure d'atteindre directement sa destination. Ceci signifie que tout nœud joue aussi bien le rôle de station que de routeur. Chaque nœud participe donc à un protocole de routage qui lui permet de découvrir les chemins existants afin d'atteindre les autres nœuds du réseau. Le fait que la taille d'un réseau ad hoc peut être très grande souligne que la gestion du routage doit être adaptée par rapport aux approches utilisées dans le routage classique. Dans le cas où le nœud destination se trouve dans la portée de communication du nœud source, le routage est évidemment trivial et aucun protocole de routage n'est nécessaire. Malheureusement, ce cas est rare dans les réseaux ad hoc. Une station source a en général besoin de transférer des données vers une autre station qui ne se trouve pas dans sa portée de communication devant ainsi faire retransmettre ses paquets par des nœuds intermédiaires. Dans la pratique, le problème de

routage est en plus compliqué par la possibilité de déplacement imprévisible de tous les nœuds concernés par le routage.

2.2 Les protocoles de routage ad hoc

2.2.1 Classification des protocoles de routage

Etant donné les limitations inhérentes aux réseaux ad hoc, la construction des routes doit être faite avec un minimum de contrôle et de consommation de la bande passante. L. Freeney propose une classification des protocoles de routage [17]. Cette classification permet d'aborder de manière structurée la multitude de protocoles de routage ad hoc existant en examinant sur base de certains critères les particularités de chacun de ces protocoles. Cette classification est effectuée suivant quatre axes :

- le modèle de communication ;
- la structure ;
- l'information au niveau des nœuds ;
- le scheduling.

Le modèle de communication isole deux classes distinctes qui sont les protocoles à canal unique et multi-canaux. Un protocole de routage à canal unique s'appuie sur le fait que la couche liaison de données est basée sur un canal logique unique. La conception d'un tel canal étant le plus souvent basée sur le principe de CSMA/CA. Les protocoles multi-canaux combinent quant à eux des assignations de canaux avec des fonctionnalités liées au routage. De tels protocoles sont généralement utilisés dans les réseaux s'appuyant sur TDMA.

Dans les protocoles à canal unique, on différencie les protocoles uniformes et non uniformes. Lorsqu'un protocole de routage est uniforme, il n'y a pas de niveau hiérarchique au sein des entités du réseau. Cela signifie que tous les nœuds répondent de la même manière aux messages de routage dans le réseau. Les protocoles non uniformes essaient quant à eux de réduire la complexité liée au routage en réduisant le nombre de nœuds participant au calcul des routes. Ce type de protocole est divisé en deux catégories : les protocoles pour lesquels chaque nœud sélectionne un ensemble de voisins avec lesquels établir les routes (neighbour selection) et les protocoles qui partitionnent la topolographie du réseau pour calculer les routes (partitioning).

Les protocoles de routage peuvent être décrits sur base de l'information obtenue au niveau de chaque nœud et/ou de l'information échangée entre ceux-ci. Les nœuds participant à un protocole orienté topographie maintiennent de l'état sur les autres nœuds du réseau. Par exemple, les protocoles

de type «link-state» sont orientés topolographie. Les nœuds utilisant un protocole orienté destination ne maintiennent de l'état que pour la topographie locale (ex : les voisins directs). Les protocoles de type «distance-vector» sont orientés destination.

Enfin les protocoles peuvent être classés suivant le moment où la source obtient les routes pour joindre des destinations. D'une part, les protocoles proactifs (table-driven) établissent les routes à l'avance en se basant sur l'échange périodique d'information sur les tables de routage. Ceci donne l'avantage de minimiser le délai pour l'obtention d'une route et de savoir rapidement si une destination est joignable. D'autre part, les protocoles réactifs (on-demand) recherchent les routes à la demande. Cette stratégie de routage s'appuyant principalement sur les mécanismes de recherche de route (route discovery) et d'entretien de la route (route maintenance) se révèle être très avantageuse dans les environnements fortement dynamiques [6].

2.2.2 DSR

Cette section présente DSR qui est le protocole de routage choisi pour effectuer les simulations. En reprenenant la classification établie par L. Freeney, on établit que DSR [7] est un protocole réactif, à canal unique, uniforme et orienté topographie. DSR utilise la technique de fowarding appelée «source routing», ce qui signifie que c'est la source qui détermine la séquence complète des nœuds à travers lesquels les paquets de données seront envoyés. Cette séquence est insérée dans l'entête du paquet à acheminer.

Principes du protocole

Pour acheminer de l'information vers une destination, l'émetteur doit d'abord construire une route en spécifiant une suite d'adresses représentant chaque nœud intermédiaire par lequel le paquet contenant l'information doit transiter pour joindre sa destination. Lorsque cette route est construite, l'émetteur peut envoyer via l'interface radio le paquet vers le premier nœud intermédiaire identifié dans la route à suivre. Lorsqu'un nœud intercepte un paquet et qu'il n'en est pas le destinaire, il le retransmet simplement vers le prochain nœud indiqué dans la route. Quand le paquet arrive à destination, les informations qu'il renferme sont délivrées aux couches supérieures du nœud destination.

Chaque entité mobile participant au réseau ad hoc maintient une cache dans laquelle il conserve les routes récemment apprises, chaque entrée de cette cache ayant une durée de vie limitée par un temporisateur. Lorsqu'un

émetteur souhaite envoyer de l'information, il vérifie d'abord sa cache pour trouver une route. Si une telle entrée est présente dans la cache, alors il utilise cette route. Dans le cas contraire, il utilisera le mécanisme de *découverte de route* (route discovery) pour trouver une route adéquate. L'information requérant une route pour être transmise sera stockée pendant un certain temps dans un buffer en attendant qu'une route soit disponible. Signalons que pendant la recherche d'une route, un nœud est toujours capable d'envoyer et de recevoir de l'information en provenance d'autres nœuds.

Etant donné la mobilité et les contraintes d'énergie des entités formant le réseau ad hoc, les routes trouvées peuvent devenir invalides. Il faut donc un mécanisme de surveillance des routes appelé *maintenance de route* (route maintenance) qui vérifie la validité des routes connues par un nœud.

Les deux paragraphes qui suivent décrivent plus en détails les deux mécanismes principaux de DSR que sont la *découverte des routes* et la *maintenance des routes.*

Découverte des routes

Vu que DSR est un protocole réactif, la demande de route n'est entreprise que s'il existe une demande pour cette route, c'est-à-dire que dans les buffers se trouve au moins un paquet qui attend d'être envoyé vers une adresse pour laquelle il n'y a pas de route.

La découverte des routes permet à n'importe quel nœud du réseau de découvrir dynamiquement un chemin vers un nœud quelconque du réseau. Une source initie une opération de découverte en diffusant un RREQ (Route REQuest) qui identifie la destination qu'elle souhaite joindre. En cas de réussite, la source reçoit un RREP (Route REPly) qui contient une séquence de nœuds à travers lesquels il faut passer pour atteindre la destination. En plus de l'adresse de la source, le RREQ contient un champ «route record», dans lequel est accumulée la séquence des nœuds visités durant la propagation du RREQ dans le réseau. Le RREQ contient également un identificateur unique de la requête. Dans le but de détecter les duplications de réceptions de la requête de route, chaque nœud du réseau maintient une liste de couples (adresse de la source, Request IDentity), des requêtes récemment reçues.

Lors de la réception d'un RREQ par un nœud N du réseau, le traitement suivant est effectué :

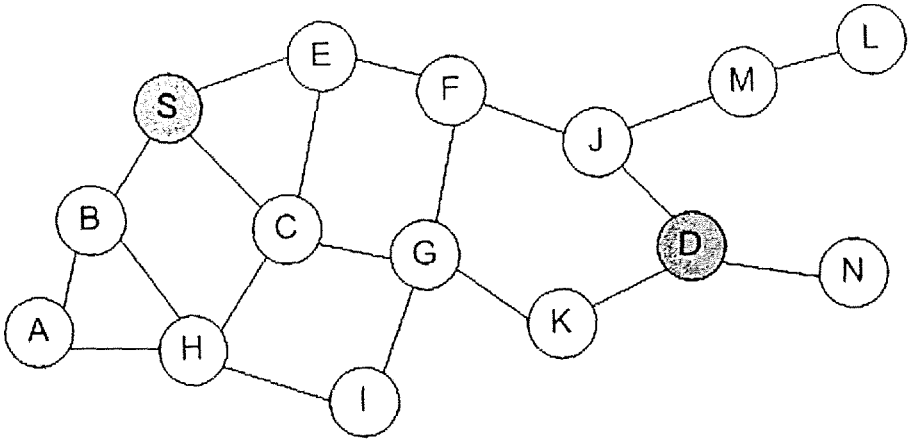
1. Dans le cas où le couple (adresse source, RID) du paquet reçu existe déjà dans la liste des requêtes récemment reçues, le paquet est ignoré ;

2. Dans le cas contraire, si l'adresse de N existe dans le route record du paquet de la requête, le paquet est ignoré ;
3. Sinon, si l'adresse de N est la même que l'adresse de la destination, alors l'enregistrement de route (contenu dans le RREQ) contient le chemin à travers lequel le paquet de la requête est passé avant d'atteindre ce nœucl. Une copie de ce chemin est envoyée dans un RREP vers l'initiateur de la recherche de route ;
4. Sinon, l'adresse de N est rajoutée dans le route record du RREQ reçu, et le paquet est rediffusé.

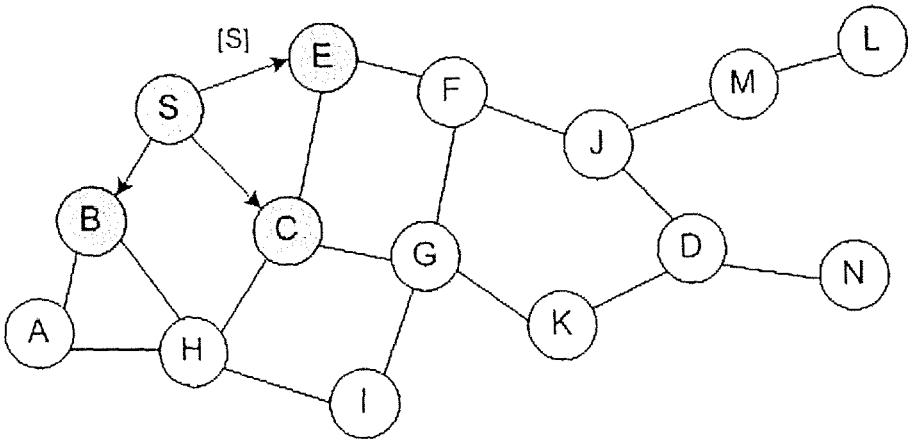
De cette manière, le RREQ est propagé dans le réseau, jusqu'à ce qu'il atteigne la destination souhaitée qui y répondra par un RREP.

Dans le but de retourner le RREP à l'initiateur de l'opération de découverte, la destination doit connaître un chemin vers cet initiateur. Pour cela, le chemin spécifié dans le route record contenu dans le RREQ peut être inversé et utilisé. Cependant, cela exige que les liens entre les nœuds participants dans le chemin soient bidirectionnels, ce qui n'est pas vérifié dans certains environnements. L'approche de piggybacking [6] peut alors être proposée dans ce genre de situation.

Les figures [16] qui suivent illustrent le fonctionnement du mécanisme de découverte des routes par DSR. Soit le réseau représenté sur la figure ci-dessous. A l'intérieur de ce réseau, le nœud S souhaite communiquer avec le nœud D. Pour que cela soit possible, S doit trouver un chemin vers D.



Pour ce faire, S envoie à ses plus proches voisins un message RREQ en incluant dans l'entête de ce message son identificateur.

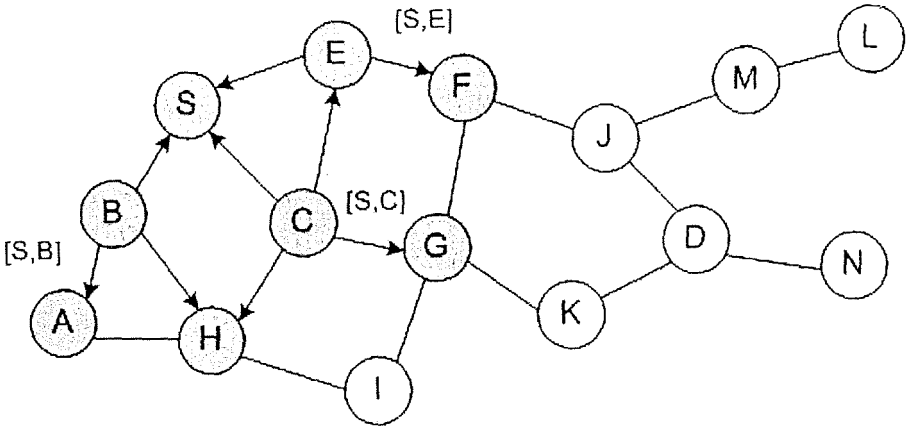


X Représente un noeud ayant reçu un RREQ

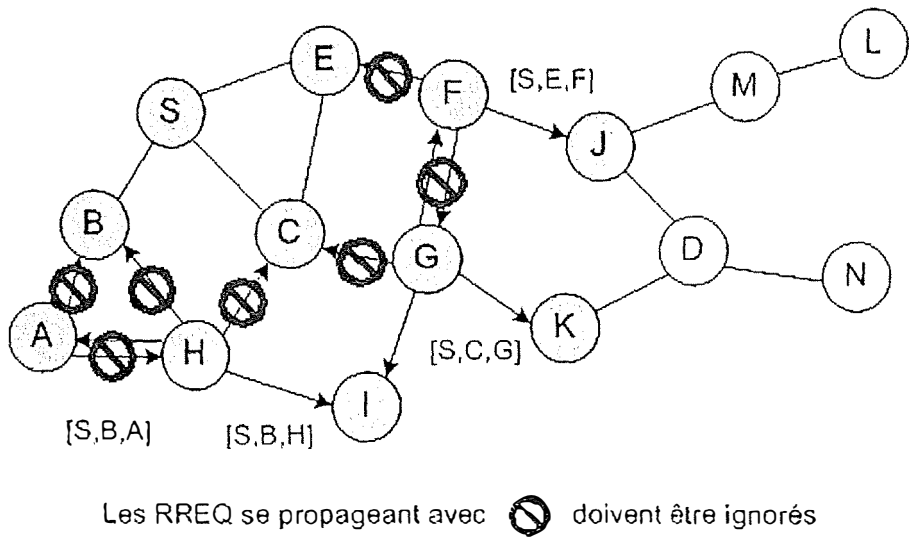
Sens de transmission des RREQ

[X, Y, Z] Liste des noeuds ajoutés dans l'entête du RREQ

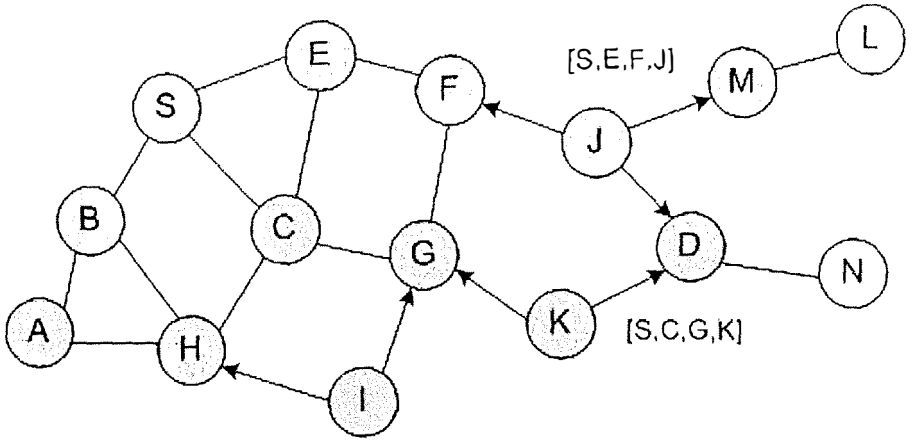
A la réception du message RREQ, chaque nœud concerné retransmet le RREQ en ajoutant à leur tour leur identifiant dans l'entête du message. De plus, chaque nœud ayant précédemment reçu un RREQ répond au nœud par lequel il a reçu ce RREQ. Remarquons que le nœud H reçoit simultanément les RREQ en provenance de B et de C. Une collision entre les deux RREQ se produit donc.



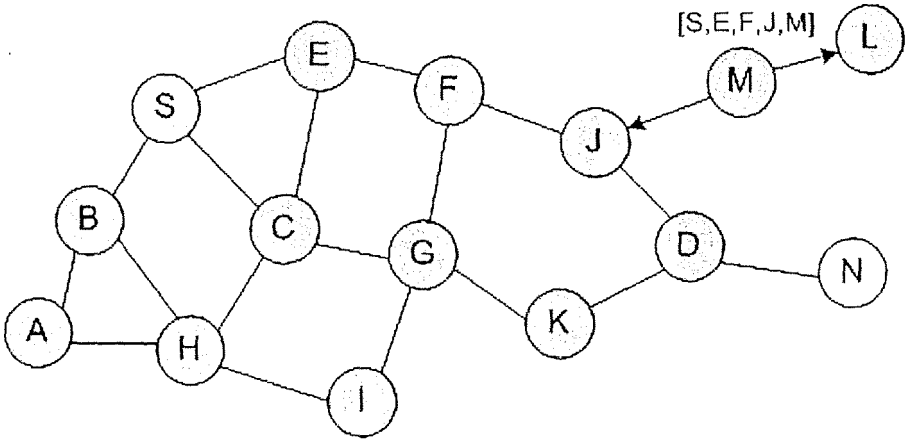
Alors que le RREQ envoyé initialement par S continue à se propager dans le réseau, des nœuds ayant déjà reçu un RREQ peuvent en recevoir à nouveau. Dans pareil cas, ils doivent ignorer ces nouveaux RREQ qui arrivent.



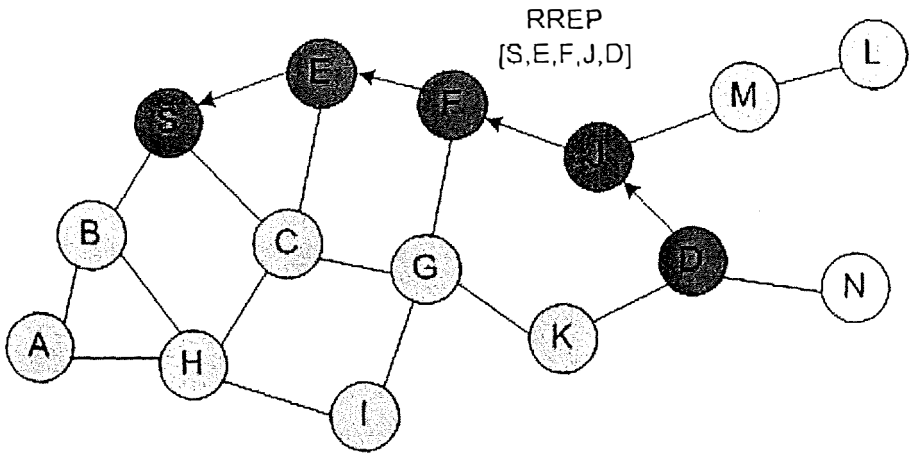
Les nœuds J et K diffusent simultanément un RREQ au nœud D.



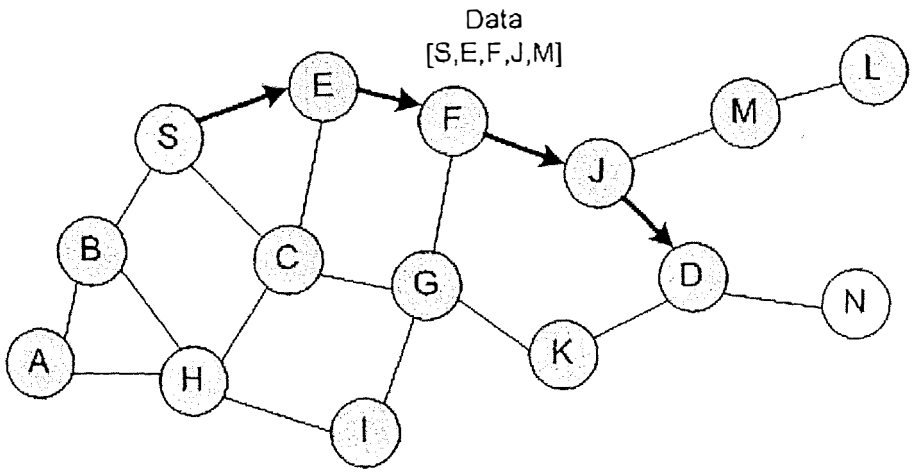
D ne doit pas rediffuser le message RREQ car il est la destination. Malgré le fait que le RREQ ait rejoint la destination souhaitée, l'exploration se poursuit dans d'autres parties du réseau (branche L,M).



D envoie alors le RREP que S attend. Pour joindre S, le RREP utilise la liste des identificateurs de nœuds (renversée pour emprunter le chemin inverse) stockée dans l'en-tête du RREQ qu'elle a reçu précédemment.



Maintenant que S dispose d'une route pour joindre D, il peut envoyer ses données par le chemin qu'il vient de découvrir.



Maintenace de la route

Afin de vérifier la validité des chemins utilisés, DSR exécute une procédure de maintenance de routes. Quand un nœud détecte un problème fatal de transmission (rupture de route), un message d'erreur de route (RERR)

est envoyé à l'émetteur original du paquet. Le RERR contient l'adresse du nœud qui a détecté l'erreur et celle du nœud qui le suit dans le chemin. Lors de la réception du RERR par la source, le nœud concerné par l'erreur est supprimé du chemin sauvegardé, et tous les chemins qui contiennent ce nœud sont tronqués à ce point-là. Par la suite, une nouvelle opération de découverte de routes vers la destination est initiée par l'émetteur.

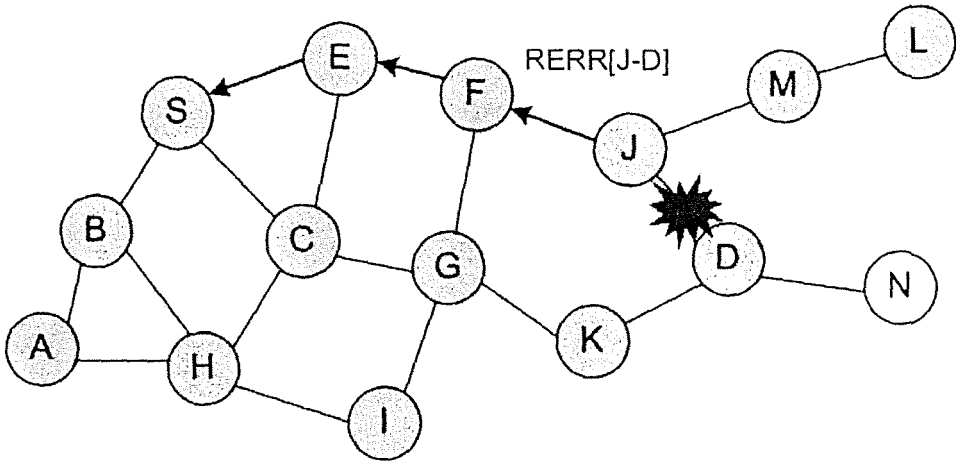


FIG. 2.1 – Détection d'une panne sur le lien J-D par le nœud J. Le nœud J propage alors vers les nœuds, dont il connait l'adresse, un message RERR reportant la panne détectée.

Chapitre 3

TCP : Transmission Control Protocol

Les protocoles de routage ad hoc fournissent une approche consistante des MANETs au niveau de la couche réseau mais ne fournissent aucune garantie de fiabilité sur le transport de bout-à-bout. TCP (Transmission Control Protocol) [30] est un protocole de la couche transport orienté connexion qui gère l'acheminement de façon fiable des informations entre deux entités terminales. TCP fournit également des mécanismes de contrôle du flux et de la congestion. C'est aussi le protocole de transport le plus utilisé dans l'Internet d'aujourd'hui. L'utilisation de TCP dans les MANETs est donc d'une grande importance si l'on veut pouvoir supporter de nombreuses applications populaires dans ce type de réseau mais aussi pour pouvoir intégrer cette technologie à l'Internet.

3.1 Les caractéristiques de TCP

TCP est un protocole de la couche transport défini dans le RFC 793 [26] qui offre un service de transfert :

- *orienté connexion* : les deux entités qui communiquent doivent établir une connexion logique pour s'échanger de l'information ;
- *fiable* en effectuant les choses suivantes :
  - les données applicatives sont découpées en fragments dont la taille est adaptée par TCP pour l'émission. L'unité d'information émise est appelée un segment ;
  - lorsqu'une extrémité reçoit des données de l'autre extrémité de la connnexion, il émet un acquittement ;
  - lorsqu'un segment est envoyé à l'intérieur du réseau, un temporisateur est déclenclé en attendant l'acquittement de réception de ce segment ;

- TCP maintient une somme de contrôle (checksum) portant sur son en-tête et sur les données. Si un segment arrive à destination avec une somme de contrôle invalide alors ce segment est jeté ;
- étant donné que les segments TCP sont transmis à l'intérieur de datagrammes IP, et que ces derniers peuvent arriver dans le désordre, les segments TCP peuvent être déséquencés. TCP est dans ce cas capable de réordonner les données si nécessaire ;

- *bidirectionnel* : les deux utilisateurs du service TCP peuvent lire et écrire les données de façon simultanée ;
- possédant des mécanismes de *contrôle du flux* : un mécanisme de fenêtre permet d'éviter à TCP d'envoyer des données au receveur qui ne possède pas suffisament d'espace mémoire pour les recevoir ;
- fonctionnant en mode *byte stream* : un flux d'octets est échangé le long de la connexion TCP entre les deux extrémités. L'émetteur et le receveur ne sont pas contraints d'envoyer et de recevoir les mêmes tailles de paquets de données. Par exemple si une extrémité écrit 10 octets, puis ensuite 20 octets et enfin 50 octets, l'autre extrémité ne sait pas dire quelle était la taille des écritures individuelles. L'autre extrémité effectuera donc par exemple 4 lectures de 20 octets pour prendre connaissance des données envoyées.

3.2 Le format des messages TCP

Le format des messages TCP est représenté à la FIG.3.1 . Les différents champs d'un en-tête TCP sont :

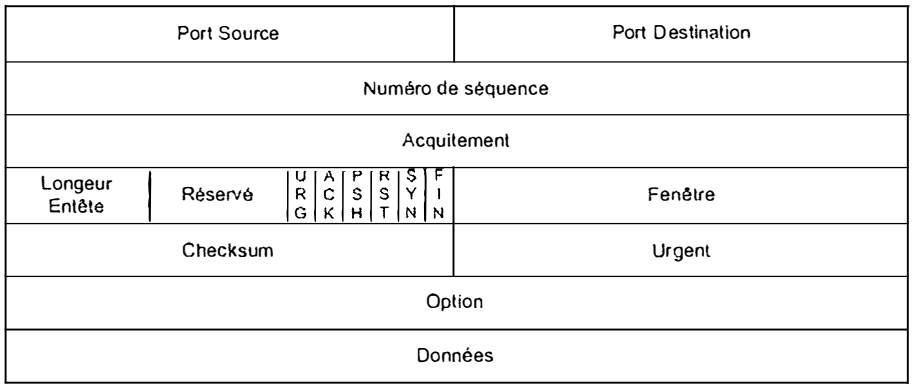


FIG. 3.1 – Format de l'en-tête TCP.

- numéros de ports : le *port source* et le *port destination* permettent d'identifier les applications s'exécutant sur les deux machines. En effet une adresse IP permet d'identifier une seule machine sur l'Internet. Or sur cette machine il peut y avoir plusieurs applications qui utilisent

le service TCP/IP. Le numéro de port vient alors s'ajouter à l'adresse IP pour identifier de façon unique une application tournant sur une machine spécifique.

- Le *numéro de séquence* qui indique le numéro de séquence du premier octet des données.
- Le champ *acquittement* qui contient le numéro de séquence du prochain octet attendu.
- Le champ *longueur d'entête* qui donne la longueur de l'en-tête.
- Le champ contenant 6 *Drapeaux* d'une taille d'un bit. Ces drapeaux permettent de définir des types de messages ainsi que la validité de certains champs. Les différents drapeaux ainsi que leur signification (lorsqu'ils sont à 1) sont comme suit :
  - URG : le pointeur de données urgentes est valide ;
  - ACK : le champ acquittement est valide ;
  - PSH : ce segment impose de délivrer toutes les données en attente à l'application ;
  - RST : demande de fermeture de la connexion à cause d'une erreur irrécupérable ;
  - SYN : ouverture de la connexion ;
  - FIN : fermeture de la connexion ;
- Le champ *fenêtre* indique à l'autre extrémité la quantité maximum de données que peut envoyer l'émetteur avant d'être obligé d'attendre des acquittements. Ce champ permet de maîtriser l'émission des paquets si le récepteur ne dispose pas suffisament d'espace mémoire. En effet, pour chaque instance de communication, TCP réserve un espace mémoire pour stocker les paquets avant que l'application locale ne les consomme. Il se trouve que les applications ne consomment pas les données au rythme de leur arrivée. Pour éviter de perdre inutilement des données, le récepteur contrôle donc le flux de données de l'émetteur en utilisant le champ fenêtre.
- Le champ *checksum* est calculé non seulement pour l'en-tête du paquet comme c'est le cas pour IP, mais aussi pour la partie données du paquet. Le checksum TCP protège donc contre les erreurs qui peuvent toucher tout le paquet TCP. Ce checksum est calculé de façon similaire à celui de l'en-tête IP.
- Le champ *pointeur message urgent* indique la position du dernier octet d'un message urgent.
- Le champ *options* permet d'échanger des données optionnelles le plus généralement pendant le phase d'établissement de la connexion. L'option la plus couramment utilisée lors de l'établissement d'une connexion est la taille maximale d'un segment (MSS : Maximum Segment Size).

3.3 Etablissement et fermeture d'une connexion TCP

Le transfert de données en utilisant TCP s'effectue en trois phases : ouverture de la connexion, transfert de données et fermeture de la connexion. Contrairement aux réseaux orientés connexion, comme X25 ou ATM, la connexion TCP est différente de la notion de circuit virtuel : les routeurs intermédiaires ne maintiennent aucun état pour la connexion TCP qui est transparente pour eux. La connexion TCP n'implique que les deux entités terminales. Pour des raisons de clarté, on appellera client l'entité TCP qui demande l'ouverture d'une connexion et serveur l'entité TCP qui répond par une acceptation ou un refus de la connexion. De façon similaire, on appellera application client un processus applicatif qui demande au fournisseur du service transport (TCP) d'ouvrir une connexion active et application serveur un processus applicatif qui demande l'ouverture d'une connexion passive. Suite à une demande d'ouverture de connexion active, TCP commence la phase d'établissement de connexion avec le serveur. Une demande d'ouverture de connexion passive signifie que l'application serveur demande l'acceptation des connexions entrantes.

Avant chaque ouverture de connexion, le processus serveur doit demander l'ouverture d'une connexion passive. Quand une connexion est ouverte, le processus serveur reçoit une indication l'informant de l'ouverture de la connexion.

Lors de la demande d'ouverture d'une connexion active, le processus client doit spécifier à TCP l'adresse IP ainsi que le numéro de port identifiant de façon unique le processus serveur. Le client envoie un segment TCP de type SYN avec un numéro de séquence égal au numéro de séquence initial (ISN : Initial Sequence Number). L'intérêt de l'ISN est de prévenir contre le risque de considérer un segment retardé dans le réseau et appartenant à une connexion antérieure comme un segment d'une connexion en cours ayant les mêmes numéros de ports que la connexion antérieure. Le serveur envoie un segment SYN en réponse à la demande de connexion. Ce segment SYN contient un acquittement égal à l'ISN du client + 1. A la réception du segment SYN, le client envoie, à son tour, un acquittement du segment SYN du serveur en acquittant l'ISN du serveur + 1 et informe le processus client de l'ouverture réussie de la connexion. De la même façon, en recevant l'acquittement du segment SYN, le serveur informe le processus serveur de l'établissement de la connexion.

A la différence de la phase d'établissement de connexion TCP qui se fait en trois étapes[1], la fermeture de la connexion TCP se fait en quatre étapes.

[1] Cette méthode d'établissement de connexion est appelé *«three-way handshake»*.

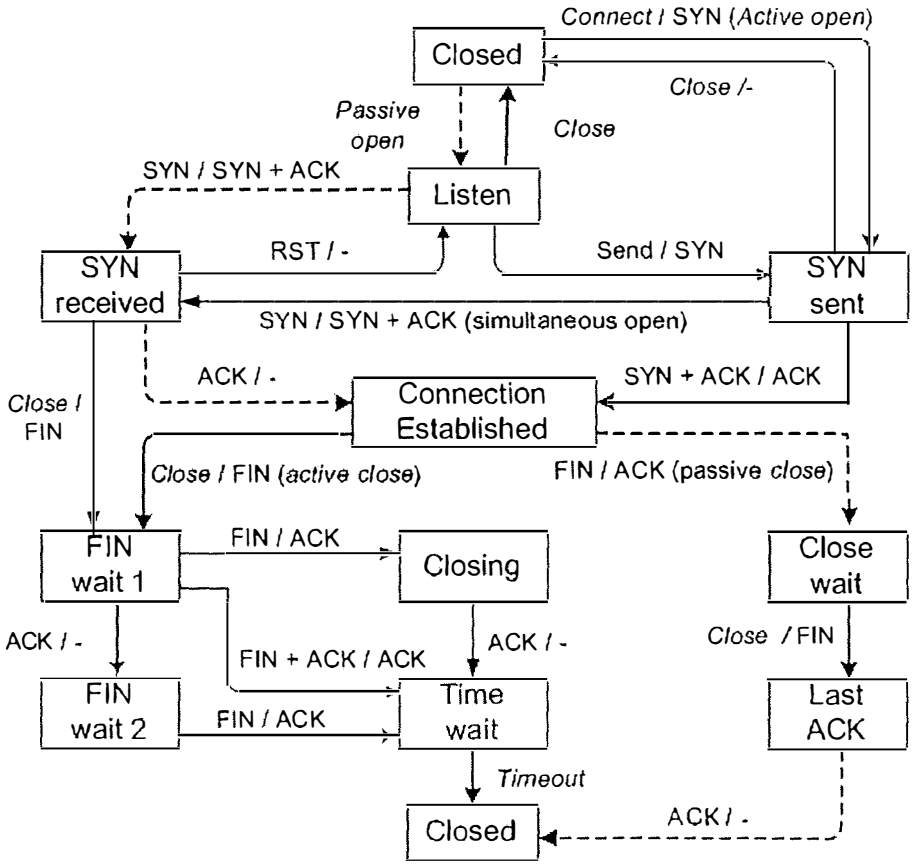


FIG. 3.2 – Représentation de l'établissement et de la fermeture d'une connexion TCP sous la forme d'un automate. Les transitions marquées par un trait plein en gras correspondent au flux normal d'exécution pour l'entité client, le cas de l'entité serveur étant représenté par les transitions en pointillés. Les autres transitions illustrent les flux alternatifs provoqués par d'autres événements pouvant intervenir lors de l'établissement ou la fermeture d'une connexion.

Comme une connexion TCP est bidirectionnelle, les processus client et serveur doivent demander la fermeture de la connexion de façon individuelle. Quand un des deux processus, client ou serveur, demande la fermeture de la connexion, le processus homologue n'ayant pas encore demandé la fermeture de la connexion peut continuer à envoyer des données. Au niveau de la couche TCP, suite à une demande de fermeture de connexion, TCP envoie un segment de type FIN et une confirmation de déconnexion est transmise au processus ayant demandé la fermeture de la connexion. Le segment FIN est acquitté par l'entité TCP homologue. Cette dernière rentre dans un état dit «semi-fermé» et peut continuer à envoyer normalement des données. Après avoir fini cette procédure d'envoi, un segment FIN est envoyé. La connexion est fermée à la réception de l'acquittement du deuxième segment FIN. Une indication de déconnexion est alors transmise au second processus. La FIG.3.2 reprend de manière complète les processus d'établissement et de fermeture d'une connexion TCP.

3.4 Echange de données

L'une des particularités de TCP est l'utilisation des numéros d'octets et non pas des numéros de paquets pour gérer les acquittements et les retransmissions. Le champ séquence indique la position du premier octet du paquet dans le flux de données alors que le champ acquittement indique le prochain numéro de l'octet attendu par le récepteur. TCP effectue le transfert d'un flux continu de données dans les deux directions en regroupant à chaque fois un certain nombre d'octets pour former des segments. En général, TCP décide quand il doit bloquer et envoyer des données en utilisant un contrôle de flux connu sous l'appellation «fenêtre glissante». Ce contrôle de flux permet d'envoyer plusieurs paquets avant de se bloquer en attente d'acquittements (Stop and Wait). Ceci permet d'améliorer le débit puisque l'émetteur n'est pas obligé d'attendre la réception d'acquittement après chaque envoi de paquets de données.

TCP fournit les moyens au receveur de contrôler la quantité de données envoyées par l'émetteur. Ceci est réalisé par le biais du champ fenêtre qui indique un intervalle des numéros de séquence qui peuvent être acceptés par le receveur.

Afin d'assurer la fiabilité, TCP effectue une retransmission des paquets perdus à l'aide de mécanismes tels que «Go-Back-N» ou «Selective Repeat». Afin de détecter ces pertes, un temporisateur est enclenché à chaque envoi d'un paquet. A la réception d'un segment qui acquitte la réception du paquet envoyé, le temporisateur est désactivé. Par contre si l'émetteur ne reçoit aucun acquittement avant l'expiration du temporisateur, le paquet en question

est retransmis.

Notons enfin que le protocole TCP utilise la notion de *Piggybacking* pour acquitter des données. Le Piggybacking permet d'optimiser l'utilisation de la bande passante lorsque l'émetteur et le receveur ont des données à émettre et consiste à ne pas envoyer un acquittement systématiquement après la réception d'un paquet de données. En recevant des données, TCP vérifie s'il a des données à envoyer. Si c'est le cas, TCP envoie un segment de données tout en mettant à jour le champ acquittement de sorte à acquitter le dernier segment reçu.

3.5 Contrôle du flux et de la congestion

L'algorithme de contrôle de flux [18] est basé sur une fenêtre et utilise une notification implicite de l'état de congestion dans le réseau. Pour ce faire TCP maintient une variable d'état appelée «Congestion Window» (cwnd) qui indique la taille de la fenêtre de congestion. La fenêtre de congestion représente la quantité maximum de données en transit dans le réseau. TCP augmente progressivement (au rythme des Round Trip Time) la valeur de cette fenêtre jusqu'à la détection d'une perte. A ce point, la source réduit la taille de la fenêtre de congestion et recommence son augmentation progressive. La croissance de la fenêtre de congestion se fait en deux phases : Slow Start et Congestion Avoidance.

3.5.1 Slow Start

Après l'établissement de la connexion, la source fixe la taille de la fenêtre de congestion à 1 segment (MSS). A chaque réception d'un acquittement, il augmente la taille de la fenêtre de congestion d'un MSS. Cet algorithme continue jusqu'à ce que la fenêtre de congestion atteigne un seuil (appelé SS-threshold). De ce comportement résulte donc un accroissement exponentiel de la fenêtre de congestion : si chaque paquet est acquitté, la fenêtre de congestion double de taille après chaque RTT . Dans les implémentations de base de TCP, le mécanisme de Slow Start est également appliqué juste après une retransmission générée par un timeout.

3.5.2 Congestion Avoidance

Après la phase de Slow Start, qui prend fin au franchissement du seuil SS-threshold, la fenêtre évolue de façon linéaire : incrémenter la fenêtre d'un MSS à chaque fois qu'une fenêtre de congestion entière est acquittée. Dans cette phase, la croissance est linéaire plutôt qu'exponentielle pour éviter de

revenir trop rapidement dans une phase de congestion. Cette phase continue jusqu'à la détection de perte de paquets[2]. A ce point, TCP met à jour la valeur du seuil, SS-threshold, à la moitié de la fenêtre de congestion.

3.6 Gestion des temporisateurs

Un des paramètres qui affectent les performances de TCP est le temporisateur de retransmission : si ce paramètre est sous-estimé, TCP peut retransmettre inutilement des segments déjà reçus et si le paramètre est sur-estimé, une perte sera détectée tardivement en résultant un temps d'inactivité de TCP assez important. Comme le protocole TCP n'a aucune connaissance des caractéristiques du réseau physique et de la charge de ce dernier, il doit ajuster dynamiquement son temporisateur de retransmission. Le RFC 793, spécifie une méthode d'estimation du timeout. L'idée consiste à calculer une moyenne des RTT mesurés et de recalculer une nouvelle valeur du timeout à chaque changement de cette moyenne. Pour ce faire, chaque connexion TCP définit les deux variables suivantes :

- *SampleRTT* : un échantillon du RTT. A chaque émission de segment, TCP enregistre l'instant de son émission. A la réception de l'acquittement correspondant, SampleRTT est calculé (différence entre l'instant de réception de l'acquittement et l'instant d'émission du segment).
- EstimatedRTT : moyenne pondérée des SampleRTT calculée ainsi :

$$a * EstimatedRTT + b * SampleRTT$$

où a et b sont deux réels positifs satisfaisants : $a + b = 1$.

La valeur du timeout est alors donnée par l'expression, conservatrice, suivante :

$$timeout = 2 * EstimatedRTT$$

.

En 1987, Karn et Partridge [4] ont introduit deux améliorations sur la gestion des temporisateurs :

- l'*exponential backoff* : TCP utilise des mécanismes de retransmission basés sur des timeouts pour gérer les pertes de paquets à l'intérieur d'un réseau. La technique de l'exponential backoff consiste à initialiser la valeur du timeout au double de sa valeur précédente à chaque fois qu'un paquet est retransmis. Etant donné que TCP interprète la perte de paquets comme un signe de congestion, ce dernier adopte en conséquence une attitude prudente en évitant de retransmettre trop rapidement des paquets sur une ligne qui lui paraît déjà chargée ;

[2]La détection de perte peut se faire de deux manières : soit par l'expiration de temporisateur, soit à la réception d'acquittements dupliqués.

- l'*estimation du RTT* : le SampleRTT ne doit pas être estimé en cas de retransmission. En effet quand un segment est retransmis tout de suite après la réception d'un acquittement, il est impossible de déterminer si cet acquittement peut être associé à la première ou à la seconde transmission du segment. D'où la nécessité de ne pas estimer le RTT pendant les retransmissions. Notons que ce problème est résolu dans le RFC1323 [5].

L'algorithme de calcul de timeout proposé dans le RFC 793 ne permettait pas de prendre en considération la variance du RTT. Intuitivement, quand la variation du RTT est très petite, il est inutile de multiplier le RTT estimé (EstimatedRTT) par deux pour avoir le timeout. Par contre quand la variation est relativement grande, le timeout ne doit pas être étroitement lié à l'estimation du RTT (EstimatedRTT). La série d'algorithmes introduite par Jacobson en 1988 inclut un mécanisme d'estimation du timeout basé sur la prise en compte non seulement du RTT estimé mais aussi de la variance du RTT. Ce nouvel algorithme garde l'estimation du RTT semblable à celle du RFC 793 mais calcule en plus une estimation de la variance (Deviation) comme suit :

$$Error = SampleRTT - EstimatedRTT$$

$$EstimatedRTT = EstimatedRTT + (\alpha * Error)$$

$$Variance = Variance + \beta(|Error| - Variance)$$

où α et β sont des réels entre 0 et 1 qui, dans la pratique, valent respectivement 1/8 et 1/4. Le temporisateur de retransmission est calculé comme suit :

$$timeout = EstimatedRTT + 4 * Variance$$

Ainsi dans le cas où la variance est très petite, le timeout est très proche du RTT estimé et dans le cas contraire la déviation domine le calcul du timeout[3].

3.7 Algorithmes de Fast Retransmit et de Fast Recovery

Les algorithmes que nous avons décrits ci-dessus constituent la brique de base pour le contrôle de trafic dans TCP. Ces algorithmes réagissent aux pertes de paquets et sont considérés comme un feedback implicite pour ajuster la taille de la fenêtre de congestion et par conséquent le trafic de

[3]Notons que les implémentations de TCP (Tahoe, Reno, NewReno et SACK) incluent le backoff exponentiel proposé par Karn et Partridge et l'estimation du timeout proposée par Jacobson.

l'émetteur. Pour avoir une indication plus rapide de la perte de paquets, Jacobson [18] a proposé l'utilisation d'acquittements dupliqués. L'idée des acquittements dupliqués est très simple : à chaque fois qu'un segment est reçu par le receveur, ce dernier doit répondre par l'envoi d'un acquittement même si d'autres acquittements avec le même numéro d'acquittement ont été envoyés. En d'autres termes, si le numéro de séquence du segment reçu n'est pas celui attendu, TCP ne peut pas l'acquitter du fait qu'un ou plusieurs autres segments sont attendus. TCP envoie alors le même acquittement que celui envoyé auparavant : d'où l'appellation acquittement dupliqué. A la réception d'un certain nombre d'acquittements dupliqués, l'émetteur TCP devine qu'il y a eu une perte de segments. Au lieu d'attendre l'expiration du temporisateur de retransmission, ce dernier retransmet le paquet perdu : d'où l'appellation Fast Retransmit (pour retransmission rapide). Notons que comme il y a un risque de déséquencement dans l'Internet et pour éviter de retransmettre inutilement des paquets, Jacobson a proposé de retransmettre le paquet estimé perdu après la réception de trois acquittements dupliqués.

L'algorithme de Fast Recovery est une optimisation de Fast Retransmit. Ce dernier, suite à une détection de perte de paquets, initialise la fenêtre de congestion à 1 et rentre dans la phase Slow-start souvent susceptible de diminuer la performance de TCP. Le recouvrement rapide est un algorithme qui s'exécute juste après le Fast retransmit du paquet perdu et peut être résumé en trois phases :

- initialiser la fenêtre de congestion ainsi que le SS-threshold à la moitié de la fenêtre de congestion à la réception du 3ème acquittement dupliqué ;
- pour chaque acquittement dupliqué reçu, envoyer un nouveau segment. Ceci permet de maintenir constant le nombre de paquets dans le réseau en envoyant un nouveau paquet chaque fois qu'un acquittement est reçu ;
- à la réception d'un acquittement du segment retransmis, quitter la phase de recouvrement rapide et rentrer dans la phase Congestion avoidance.

La FIG 3.3 montre l'évolution de la fenêtre de congestion TCP en fonction du délai aller-retour (RTT).

3.8 TCP/SACK

Certaines implémentations de TCP[4] utilisent une technique de retransmission appellée Go-Back-N. Dans Go-Back-N, après la détection d'une perte, l'émetteur retransmet non seulement le segment perdu mais aussi des seg-

[4]TCP/Tahoe et TCP/Reno par exemple.

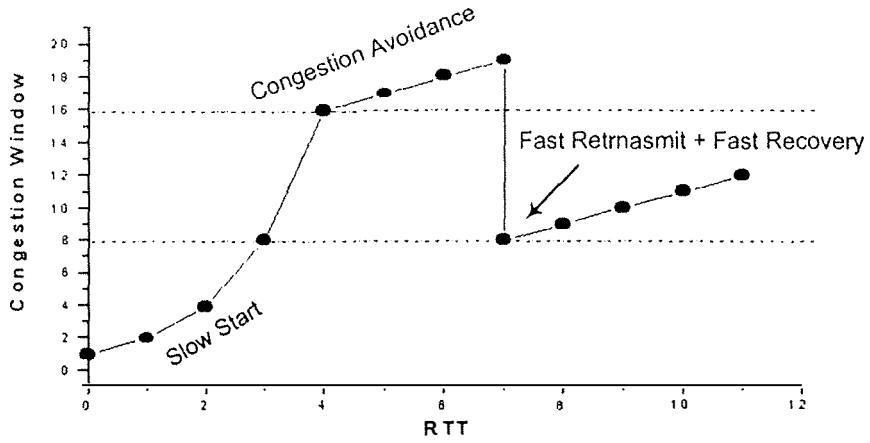


FIG. 3.3 – Evolution de la fenêtre de congestion en fonction du délai aller-retour.

ments qui lui succèdent. L'inconvénient d'une telle stratégie est que les paquets retransmis peuvent être déjà reçus par le réceveur. Pour remédier à ce problème il est possible d'utiliser les «acquittements sélectifs». Le principe des acquittements sélectifs est simple et consiste à reporter dans les acquittements les blocs contigus de segments reçus par le receveur. De cette façon, l'émetteur ne retransmet alors que les segments perdus.

TCP/SACK [3] inclut Slow Start, Congestion Avoidance, Fast Retransmit et Fast Recovery. Pendant la phase de Fast Recovery, TCP/SACK maintient deux nouvelles variables : *pipe* qui est une estimation du nombre de paquets dans le réseau et *ndup* qui comptabilise le nombre d'acquittements dupliqués reçus. En plus, l'émetteur garde en mémoire une image des blocs contigus reçus par le réceveur. Au début de la phase de Fast Recovery, les initialisations suivantes sont faites : $pipe = cwnd - ndup$ et $cwnd = \frac{cwnd}{2}$. A chaque fois qu'un acquittement est reçu la variable pipe est incrémentée. Quand pipe est inférieure à cwnd, TCP est autorisé à transmettre un segment. Pour ce faire, la liste des blocs contigus est consultée pour déterminer le prochain segment à retransmettre. Dans le cas où tous les segments reportés perdus sont retransmis, TCP est autorisé à transmettre un nouveau paquet. Notons que pour conserver le nombre de paquets dans le réseau constant durant la phase de recouvrement rapide, un paquet n'est transmis que si la condition suivante est satisfaite : $cwnd \leq pipe$. En plus, à chaque transmission, la variable pipe est incrémentée. L'émetteur sort de la phase de Fast Recovery quand tous les paquets transmis juste avant la phase de Fast Recovery sont acquittés.

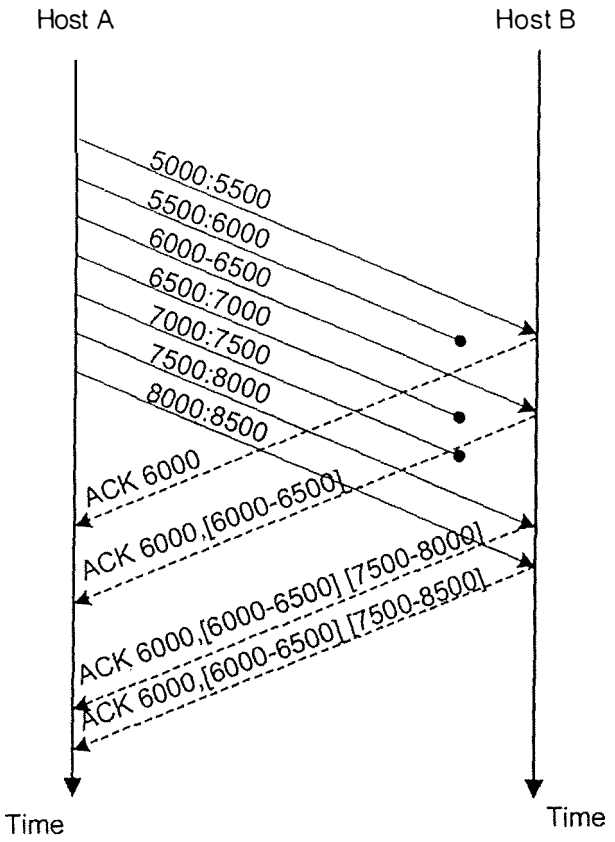


FIG. 3.4 – Exemple d'acquittement utilisant l'option SACK.

L'utilisation des acquittements sélectifs améliore considérablement les performances quand le réseau présente, au niveau physique, un taux de pertes élevé (e.g. liaisons satellite, réseaux sans fil...). Notons enfin que la variante SACK est la version de TCP qui est actuellement implémentée au niveau des sytèmes terminaux. L'utilisation de l'option SACK est négociée lors de l'établissement de la connexion. Ceci permet de garder une compatibilité avec les versions antérieures de TCP.

3.9 TCP dans les MANETs

Dans un réseau de type MANET, un protocole tel que TCP se heurte à un certain nombre de problèmes [23] tels que :

- *un taux d'erreur de transmission élevé* : dans les MANETs le taux de pertes sur un canal de transmission est élevé. Lorsque TCP ne reçoit pas d'acquittement, il effectue un exponential backoff sur le tempori-

sateur. Cette solution de réduire le nombre de retransmissions n'est certainement pas la meilleure car dans ce cas, les pertes de paquets ne sont pas la conséquence d'une congestion à l'intérieur du réseau ;

- *(Re)calcul fréquent des routes* : étant donné que chaque entité du réseau ad hoc est animée d'une certaine mobilité, les routes risquent d'être recalculées fréquemment et la connexion supposera, pendant le temps nécessaire au calcul des routes, qu'il existe de la congestion à l'intérieur du réseau et ce, jusqu'à ce que une nouvelle route soit trouvée. Durant cette période, les algorithmes de congestion vont donc être activés. En plus, les paquets émis par l'entité source risquent d'être jetés par les entités intermédaires qui ne sont pas capables de les acheminer correctement à cause de l'absence de route valide pour joindre la destination, ce qui aura aussi pour conséquence une augmentation du délai aller-retour (Round Trip Time). Tout cela provoquant de manière générale une dégradation des performances de TCP. Ce problème est d'autant plus critique si le temps de calcul des routes est relativement grand à l'échelle de la variable d'état RTO (Retransmission TimeOut) maintenue par l'entité émettrice du service du transport ;
- toujours à cause de la mobilité, des *partitions* peuvent se créer dans le réseau. Ce phénomène de partition trouve son origine dans l'absence d'infrastructure centralisée caractérisant les réseaux ad hoc. Si l'émetteur et le receveur se trouvent dans des partitions distinctes, tous les paquets envoyés par l'émetteur seront jetés à l'intérieur du réseau par défaut de route joignant la destination. De plus, pour chaque retransmission échouée, le RTO est doublé (exponential backoff). Si cet état partitionné du réseau dure relativement longtemps (plusieurs fois RTO) alors, plusieurs retransmissions infructueuses du même paquet seront effectuées avec comme résultat final, si le nombre de retransmissions est trop élevé, une fermeture de connexion ;
- certains algorithmes de routage[5] maintiennent au niveau de l'entité source plusieurs routes vers une destination particulière, dans le but d'éviter le recalcul fréquent des routes. Cette stratégie provoque un effet de bord. En effet, un grand nombre de paquets peut potentiellement arriver hors séquence chez le receveur, obligeant ce dernier à reséquencer ces paquets.

On trouve dans la littérature plusieurs améliorations de TCP adaptées aux environnements ad hoc [13, 15, 21] et proposant des mécanismes pour résoudre ces différents problèmes.

[5] par exemple TORA

Chapitre 4

Résultats expérimentaux

Ce chapitre reprend les résultats obtenus lors des séries de manipulations effectuées avec le simulateur ns2[1]. Le but de ces expériences est de caractériser les performances du service de transport TCP dans un environnement MANET. A cette fin, plusieurs ensembles de scénarii ont été réalisés : le premier présente une topographie statique (c'est-à-dire où tous les participants sont fixes) de type string [11] . Le second est une variante du premier où l'on garde une topographie représentant un arrangement linéaire de nœuds statiques mais en introduisant cette fois un émetteur se déplaçant parallèlement à cette rangée de nœuds fixes. Le troisième scénario représente un ensemble de nœuds fixes placés à égale distance les uns des autres sur un cercle. Un émetteur se déplace alors autour du cercle, sur un cercle concentrique de rayon supérieur à celui sur lequel sont disposées les entités fixes. Enfin, le dernier scénario reprend des «nuages» de nœuds mobiles se déplaçant suivant un mouvement aléatoire. Chacun de ces scénarii est pensé pour mettre en évidence des éléments qui permettent l'évaluation des performances du service de transport TCP. Signalons également que les différents scénarii envisagés sont volontairement simplifiés afin de pouvoir vérifier la validité des résultats obtenus.

Pour chaque ensemble de manipulations les paramètres suivants ont été choisis : TCP/Sack pour le protocole de transport et DSR pour le protocole de routage ad hoc. Cette implémentation de TCP fut exécutée en utilisant des paquets d'une taille de 512 et 1460 bytes qui sont deux valeurs proches des principales tailles caractéristiques de paquets échangés dans l'Internet d'aujourd'hui [1]. De plus, cette taille est laissée constante durant toute la durée de la simulation. La taille maximum de la fenêtre fut initialisée aux valeurs 32 KB et 64 KB qui sont des valeurs communes se retrouvant dans les implémentations de TCP [30]. Précisons tout de même que le simulateur ns2 représente la taille de la fenêtre TCP en nombre de paquets et que donc

[1] Une présentation du simulateur est disponible en annexe.

les valeurs d'initialisation de la fenêtre citées ci-dessus doivent être converties en nombre de paquets, ce qui donne une taille de fenêtre de 64 (22) et 128 (44) pour un paquet ayant une taille de 512 (1460) bytes. Dans chaque scénario, le transfert de données est réalisé en effectuant un transfert infini lors d'une session FTP. Le fait de recourir à un transfert infini de données permet de s'affranchir de certains effets de bord inhérents à la simulation et aussi de minimiser l'impact sur le débit de mécanismes tels que slow-start.

Pour chacune de ces manipulations, le débit d'une seule connexion TCP a été évalué. Le débit est défini ici comme le nombre total de bits reçus à chaque seconde par le receveur pendant la durée de vie de la connexion TCP. Il s'agit donc d'une mesure du goodput de la connexion TCP. Suivant les scénarii, d'autres paramètres seront également observés tels que le taux de perte de la connexion, le délai de bout à bout ou encore l'évolution de la fenêtre de congestion. Dans toutes les simulations envisagées, aucun trafic de base n'est mis en concurrence avec la connexion TCP. Les observations faites sont donc uniquement liées à cette dernière.

4.1 Topographie statique

Le but de cette première manipulation est d'observer le débit d'une connexion TCP dans un réseau ad hoc où tous les nœuds sont statiques. Pour cela, un transfert infini de données (session FTP) a été effectué dans un réseau linéaire constitué de 2 à 11 nœuds équidistants les uns des autres, l'émetteur et le receveur se trouvant à chaque extrémité (FIG. 4.1).

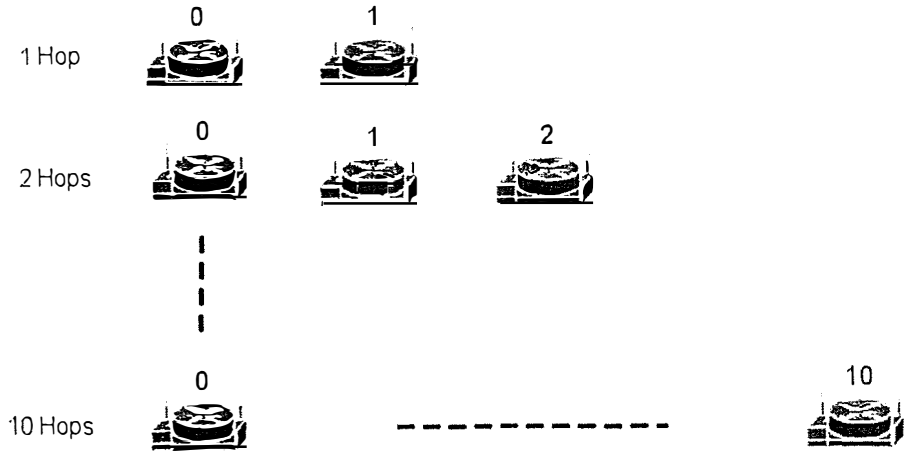


FIG. 4.1 – Topographie utilisée dans ce premier ensemble de manipulations. L'émetteur et le receveur se trouvent aux extrémités de la chaîne.

L'intérêt d'un tel réseau est que son aspect statique permet de minimiser la charge générée par le protocole de routage principalement lorsque celui-ci doit reconfigurer sa table des routes. Cela permet également d'obtenir en quelque sorte une borne supérieure sur le débit que l'on peut espérer atteindre en fonction du nombre de sauts à effectuer. Certains auteurs utilisent ces valeurs afin de déterminer un «débit idéal» servant à borner supérieurement leurs mesures de débit [9]. De plus, les résultats obtenus avec ce scénario serviront de base pour les autres manipulations.

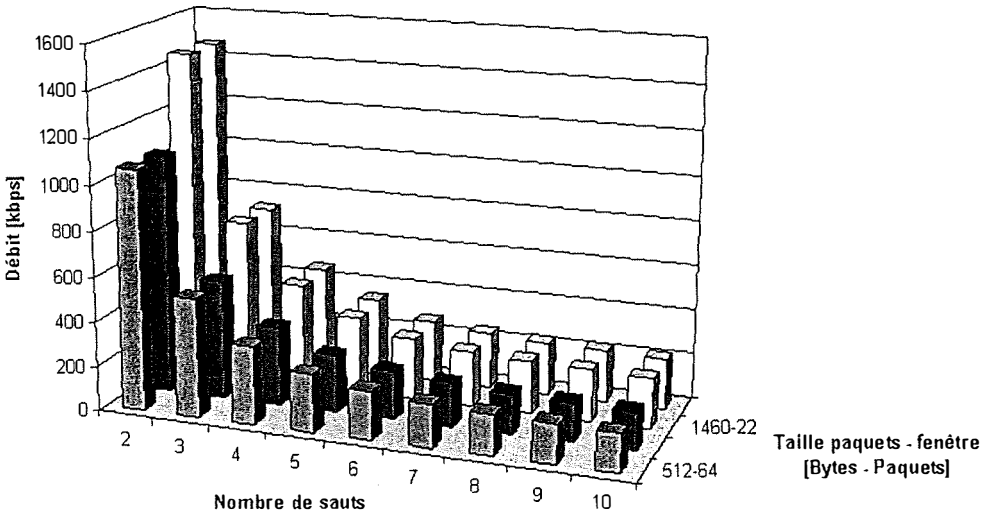


FIG. 4.2 – Mesure du débit dans un réseau linéaire statique.

Les différents débits mesurés sont résumés à la FIG. 4.2. Les chiffres indiqués représentent une moyenne sur différentes simulations effectuées en faisant varier l'espacement entre les nœuds (de 130 à 250 m). Pour chaque série de simulations, l'espace entre deux nœuds adjacents était le même, calculé de façon à ce qu'un nœud de la chaîne ne puisse communiquer qu'avec ses deux plus proches voisins. Cette considération permet de supposer que tous les nœuds de la chaîne adopteront ainsi le même comportement.

On remarque tout d'abord que la taille des paquets a une influence sur le débit : un paquet d'une taille de 1460 bytes donne un débit plus important qu'un paquet d'une taille de 512 bytes. En effet, les couches TCP, IP et MAC du nœud ajoutent un en-tête de quelques bytes. Comme le nombre de bytes de contrôle ainsi rajouté est constant, une augmentation de la taille

des paquets fait croître le rapport entre la quantité de données utiles et la taille de l'en-tête. Par conséquent, le débit utile est plus important. De plus, la différence de débit est également marquée par le délai supplémentaire introduit lors de la transmission des paquets au niveau de la couche MAC, suite au fonctionnement du protocole IEEE 802.11 (RTS, CTS et ACK).

Au niveau de la taille de la fenêtre, il semble n'y avoir aucune influence sur le débit. En effet, étant donné que le réseau est statique, les pertes sont nulles et donc la taille de la fenêtre, pour les valeurs qui ont été choisies, n'influence pas le débit pour cette manipulation.

On remarque également que quand le nombre de sauts double, le débit diminue de moitié. Cependant, au fur et à mesure que le nombre de sauts augmente, le débit tend à se stabiliser. Ce comportement trouve une explication en examinant l'interaction entre la solution retenue par IEEE 802.11 pour résoudre le problème de la station cachée et la politique de TCP pour la gestion des fenêtres. Considérons les nœuds i, i+1 et i+2. Si le nœud i veut envoyer de l'information vers i+2, il doit tout d'abord «gagner» le droit d'accès au canal. Quand le canal lui est alloué, il peut envoyer ses informations (en nombre limité) vers i+1. A ce moment, i doit attendre que i+1 achemine les données stockées dans son buffer vers i+2. Et ce n'est que quand i aura reçu un acquittement de la part de son receveur qu'il pourra avancer sa fenêtre TCP d'une unité. Pourtant, lorsque le nombre de sauts augmente (plus de 6), le débit tend à se stabiliser. La raison en est qu'à partir d'un certain nombre de sauts, un certain parallélisme s'installe . En effet, dès que la chaîne devient suffisamment grande, chaque extrémité peut transmettre ou recevoir des données simultanément.

4.2 Topographie linéaire avec nœud mobile

Dans cette manipulation, un scénario de base identique au précédent est conservé à la différence que cette fois, l'émetteur se déplace parallèlement au-dessus du réseau linéaire, le receveur restant fixe à l'extrémité gauche de la partie linéaire (FIG. 4.3).

Le but de cette expérience est d'observer le comportement d'une connexion TCP lorsqu'on introduit une faible mobilité (un seul nœud se déplace). Le fait d'éloigner l'émetteur du receveur ajoute des intermédiaires entre ces deux derniers.

La FIG. 4.4 donne l'évolution du débit de la connexion TCP en fonction du temps. On remarque que le débit diminue au cours du temps, ce qui est la

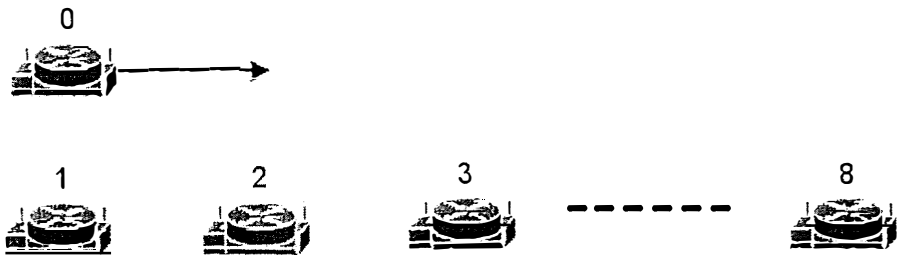


FIG. 4.3 – Topographie linéaire avec un émetteur se déplaçant au-dessus de la partie linéaire statique. L'émetteur et le receveur sont représentés respectivement par les nœuds 0 et 1.

traduction de l'éloignement progressif de l'émetteur par rapport au receveur et donc de l'ajout d'intermédiaires. De plus, la valeur du débit mésurée à chaque palier correspond aux valeurs mesurées dans la première manipulation. Sur le graphique est également reportée la courbe théorique du débit. Cette courbe fut calculée en utilisant un modèle simple de prédiction [27].

On remarque que la courbe mesurée expérimentalement correspond à la courbe théorique, sauf en quelques points où l'on assiste à des chutes abruptes de débit. Comme on peut le voir sur la FIG. 4.6, ces chutes abruptes de débit sont dues à des pertes de segments TCP. Au niveau de chaque palier, des paquets DSR sont échangés. Ces paliers correspondent en fait à l'ajout d'un intermédiaire suite au déplacement de l'émetteur, cette situation introduisant un défaut de route. Suite à cette perte de route, l'émetteur doit rechercher à nouveau un chemin pour pouvoir continuer à envoyer ses données vers la destination. Il y a donc dans le réseau une circulation de paquets DSR lorsque les routes sont brisées, ceci étant conforme à la nature réactive du protocole de routage DSR. Lorsque le nombre de nœuds intermédiaires augmente et que la longueur de la route atteint 5 sauts, on obseve des chutes de débit régulières jusqu'à des valeurs nulles ou proches de zéro. Etant donné que la connexion TCP étudiée est la seule source de trafic dans le réseau, cette situation n'est pas attendue. Cette observation est en fait une autre conséquence de l'interaction entre TCP et la couche MAC IEEE 802.11. [14] ont égalemement remarqué ce phénomène et le qualifient «d'instabilité de TCP». Ces oscillations de débit résultent de tentatives infructueuses d'accès au canal par un nœud pour joindre le nœud adjacent. En effet, quand un nœud ne parvient pas à accéder au canal de transmission (IEEE 802.11 permet 7 essais pour y accéder), il jette tous les paquets à acheminer vers la destination qui se trouvent dans ses buffers et reporte un défaut de route, cela provoquant la recherche d'une nouvelle route pour joindre la destination. Pendant ce laps de temps, aucun paquet TCP ne peut être acheminé

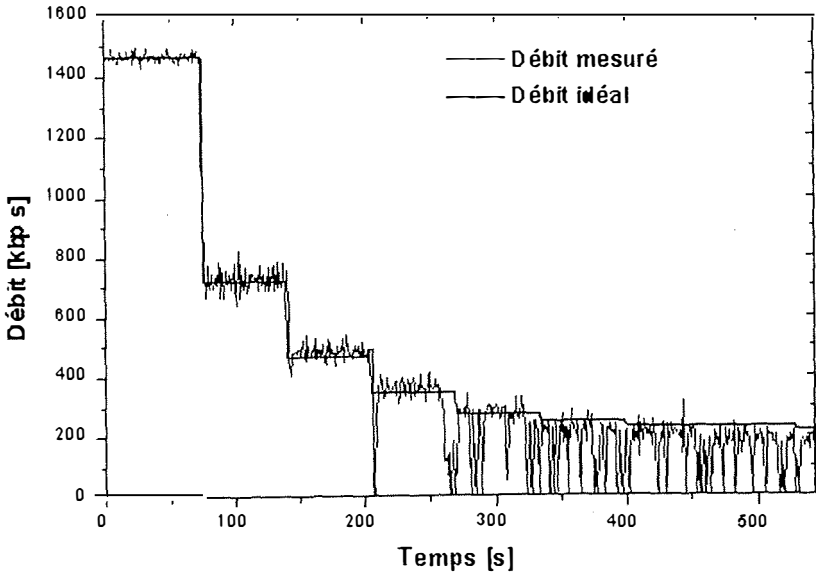


FIG. 4.4 – Evolution du débit de la connexion TCP pendant la simulation. Voici les principaux paramètres de la simulation : taille des segments : 1460 bytes ; taille de la fenetre : 64 KB ; vitesse de déplacement 10 m/s ; distance moyenne entre les nœuds : 130m.

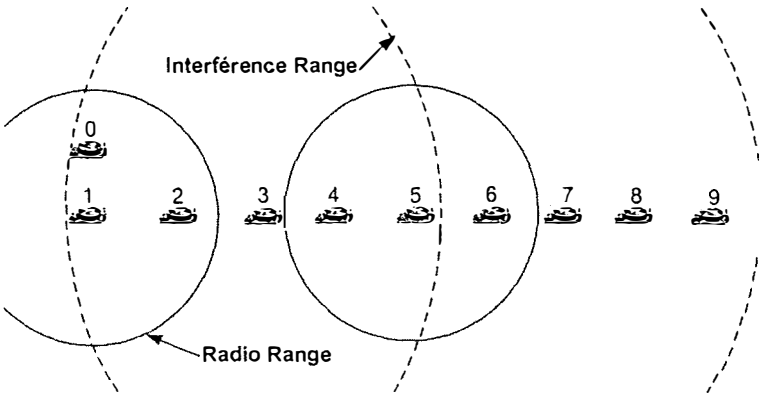


FIG. 4.5 – Portée radio et portée d'interférences illustrées pour les nœuds 0 et 4 de la chaîne linéaire.

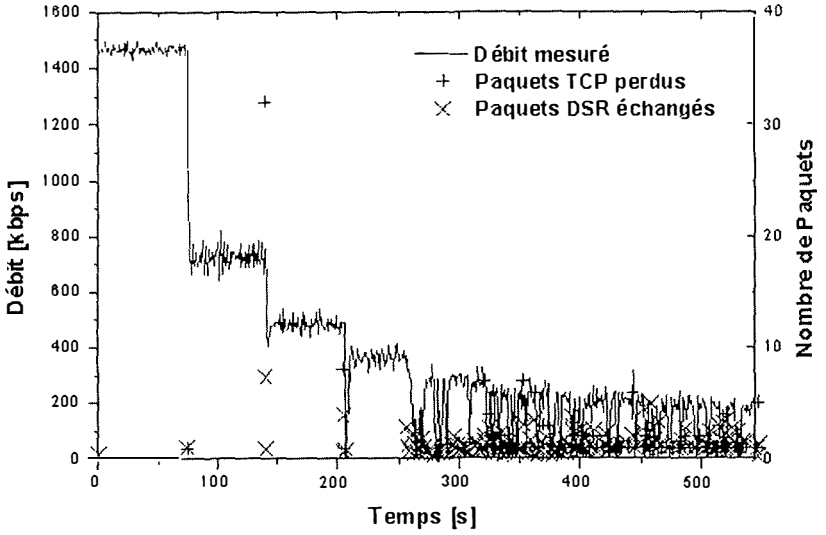


FIG. 4.6 – Evolution du débit de la connexion avec le nombre de segments TCP perdus et le nombre de paquets DSR échangés.

vers la source et le débit est donc fortement réduit. Cela explique la grande proportion de paquets DSR échangés tout au long de la seconde partie de la simulation.

Un autre phénomène qui vient lui aussi renforcer celui exposer ci-dessus sont les interférences au niveau des interfaces radio [12]. En effet, si la portée radio effective, c'est-à-dire la distance jusqu'à laquelle l'émetteur peut capter correctement les signaux radio, est de 250 mètres, la portée d'interférence, c'est-à-dire la distance jusqu'à laquelle deux nœuds peuvent interférer, est de 550 mètres. Donc, lorsque l'émetteur est à l'extrémité de la chaîne, peu de nœuds interféreront. Alors qu'au fur et à mesure que l'émetteur parcourt la chaîne linéaire, de plus en plus de nœuds interféreront ensemble. Par exemple, sur le graphique de la FIG.4.5, qui reprend un scénario de cette manipulation, les nœuds sont espacés d'environ 130 mètres. Ce qui veut dire que le nœud 1 aura dans sa portée radio uniquement les nœuds 0 et 2 mais pourra interférer avec les nœuds 3, 4 et 5. Le nœud 5, quant à lui, aura dans sa portée radio les nœuds 4 et 6 mais pourra interférer avec les nœuds 0, 1, 2, 3, 4, 6, 7, 8 et 9.

Regardons maintenant un autre élément important qui est la mesure du délai. Ce dernier peut être estimé à l'aide du Round Trip Time (RTT). Pour rappel, le RTT représente le temps écoulé entre l'envoi d'un paquet et la ré-

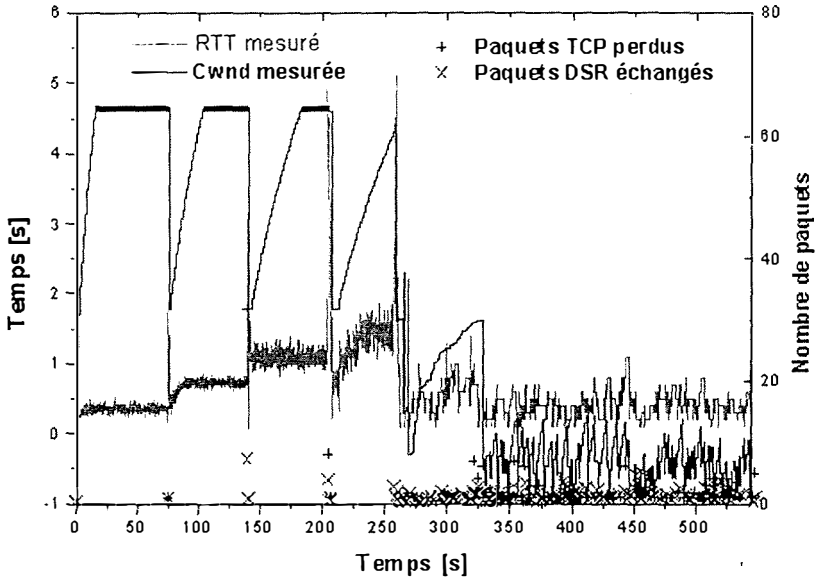


FIG. 4.7 – Evolution temporelle du délai aller-retour des paquets tout au long de la connexion et variation de la taille de la fenêtre de congestion. Le nombre de segments TCP perdus ainsi que le nombre de paquets DSR échangés, déjà présentés sur la FIG.4.6 ont également été reportés.

ception de l'acquit correspondant. La FIG.4.7 donne l'évolution temporelle du RTT. Au début de la simulation, le délai aller-retour augmente de manière régulière au fur et à mesure que le débit dinimue. Cependant, à partir de 260 secondes, lorsque la route suivie par les paquets de données atteint une longueur de cinq nœuds intermédiaires, on assiste à une diminution du délai aller-retour alors qu'en toute logique on devrait s'attendre à une persistence de l'augmentation de cette variable. Il est possible que ce phénomène soit une conséquence du parrallélisme introduit au niveau de la couche MAC tel que nous l'avions annoncé dans le cadre du premier scénario (voir page 37).

La fenêtre de congestion, quant à elle, une progression régulière en début de simulation : après avoir atteint son maximum, elle est réduite de moitié lorsqu'une perte de paquets survient. Cette situation se répète jusqu'à environ 260 secondes. Après cet instant, elle adopte un comportement différent, à l'instar du délai aller-retour. En effet, à partir de cet instant, les pertes de paquets se faisant plus fréquentes, la taille de la fenêtre jusqu'à la fin de la

simulation ne dépasse plus les 10 unités.

4.3 Topographie en cercle avec nœud mobile

Dans cette topographie, les 9 nœuds composant la chaîne linéaire ont été placés sur un cercle, chaque entité étant à égale distance des autres. La station mobile se déplace quant à elle sur un cercle concentrique mais de diamètre supérieur (FIG.4.8).

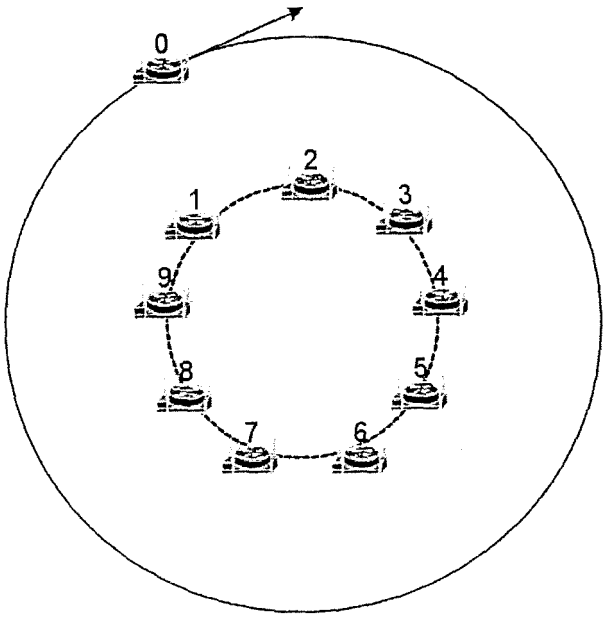


FIG. 4.8 – Topographie circulaire élaborée pour l'étude de l'évolution du débit et du taux de perte en fonction de la vitesse de déplacement du nœud mobile. Les nœuds 0 et 1 désignent respectivement l'émetteur et le receveur.

Le but de cette manipulation est de mesurer l'influence de la vitesse de déplacement du mobile sur le débit. Cette étude aurait très bien pu être réalisée à l'aide du scénario précédent, mais le principal problème de la chaîne linéaire est qu'elle ne permet d'effectuer qu'un déplacement fini. En effet, étant donné que la chaîne possède une longueur finie, le nœud mobile est forcé de terminer son mouvement au bout de cette chaîne. Ceci peut être gênant lorsque, par exemple, on effectue une simulation en initialisant la vitesse de l'entité mobile à 40 m/s. A cette vitesse, il ne faut donc qu'une poignée de secondes à l'entité mobile pour rejoindre l'autre extrémité de la chaîne, ce qui du point de vue des résultats obtenus est largement insuffisant

pour extraire des informations pertinentes. L'avantage de remplacer la configuration linéaire par une configuration circulaire est que dans ce cas il est permis au nœud mobile de circuler autour du cercle de nœuds fixes autant de temps qu'il le souhaite.

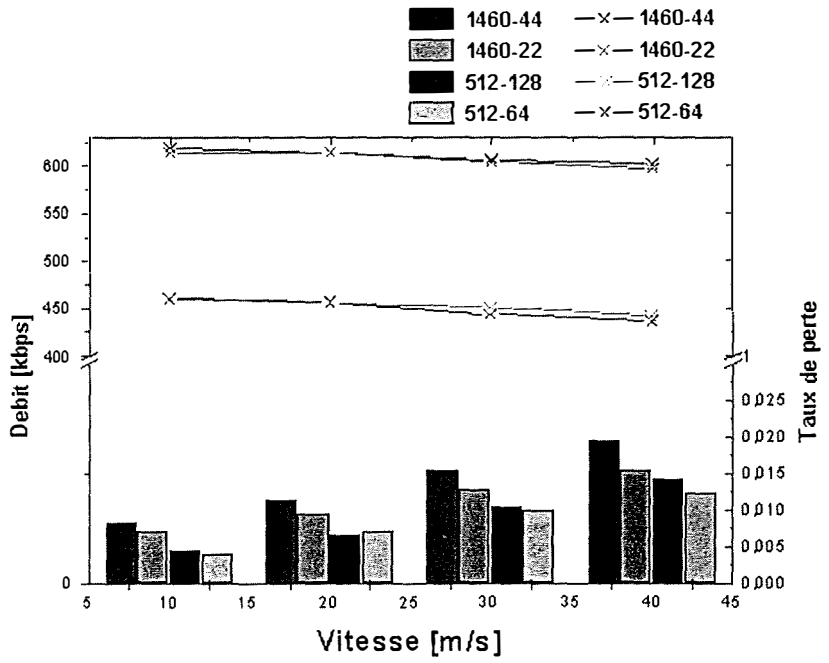


FIG. 4.9 – Evolution du débit (partie supérieure) et du taux de perte de paquets (bâtonnets) en fonction de la vitesse de déplacemement du nœud mobile.

La FIG.4.9 montre l'évolution du débit et du taux de perte pour plusieurs vitesses de déplacement de l'entité mobile. Signalons d'abord que tous les paquets perdus sont des paquets jetés par les nœuds intermédiaires à la suite d'un défaut de route. On peut voir ensuite que, globalement, le taux de perte augmente avec la vitesse alors que le débit a tendance à plutôt diminuer. En effet, dans le scénario tel qu'il a été envisagé, lorsque la vitesse du nœud augmente, les ruptures de route sont plus fréquentes et donc les pertes consécutives à ces défauts de route augmentent également. Cette augmentation du taux de perte entraîne donc une baisse de débit de quelques kbps.

On remarque aussi que la taille de la fenêtre a une influence sur le taux de perte. Ceci s'explique par le fait que lorsque la taille de la fenêtre augmente,

plus de paquets peuvent être envoyés dans le réseau ; et dès lors, quand une rupture de route survient plus de paquets seront jetés. Ajoutons enfin que le fait d'utiliser une implémentation de TCP supportant l'option Sack aide sans doute à réduire cette augmentation du taux de perte car cette variante de TCP est conçue pour limiter le nombre de retransmissions en ne faisant retransmettre que les paquets manquants.

4.4 Mouvements aléatoires

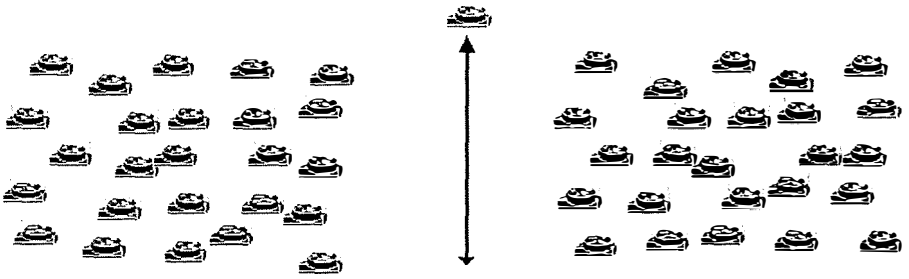


FIG. 4.10 – Topographie aléatoire.

La dernière topographie étudiée est un peu plus complexe que les précédentes : dans ce scénario, deux groupes de 25 stations sont animés d'un mouvement aléatoire. Entre ces deux groupes, il existe une distance supérieure à la portée radio des entités mobiles. L'émetteur et le receveur se trouvent dans des groupes dinstincts, ils ne peuvent donc dialoguer. Pour leur permettre de s'échanger de l'information, un nœud solitaire décrit une trajectoire rectiligne à mi-distance de chaque groupe de nœuds, pouvant ainsi relayer le trafic d'un groupe à l'autre. Le trafic pourra être ainsi acheminé tant que le nœud solitaire restera à portée des deux groupes. Lorsque ce ne sera plus le cas, une partition sera à nouveau créée. Après un certain temps, ce nœud isolé fera demi-tour pour revenir à sa position initiale. Sur le chemin du retour la connectivité entre les deux groupes de nœuds pourra donc être rétablie et les informations pourront à nouveau être acheminées vers le receveur.

Le but de cette manipuplation est d'observer le comportement de TCP face à la création subite de partitions, ce qui est un événement fréquent dans les environnements de type MANET.

La FIG.4.11 montre l'évolution du débit pendant la simulation. Ce débit reste constant à une valeur approximative de 200 kbps. La longueur moyenne de la route suivie est de 6 sauts. La valeur du débit mesuré est cependant plus

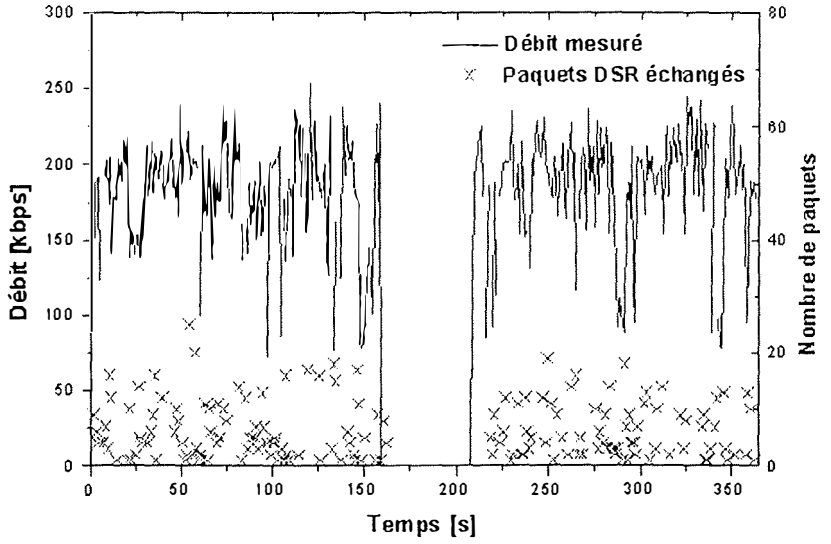


FIG. 4.11 – Evolution temporelle du débit et échange des paquets DSR durant la simulation. Paramètres de la simulation : taille des paquets TCP : 1460 Bytes; taille de la fenêtre : 64 KBytes; durée de la simulation : 365 secondes; écartement initial entre les nœuds d'un groupe : 150 m; distance entre les deux groupes : 300 m ; vitesse moyenne des nœuds à l'intérieur d'un groupe : 2 m/s avec un temps de pause moyen de 20 secondes; vitesse du nœud solitaire : 10 m/s .

faible (à nombre de sauts égal) que dans le cas de la chaîne linéaire statique. Ceci est la conséquence de la surcharge occasionnée par le trafic des paquets de contrôle DSR. Pour rappel, un des objectifs de la topographie en chaîne linéaire était de minimiser la charge générée par les paquets de routage.

On remarque également que la partition se crée à l'instant 160 secondes et que la partition persite pendant environ 50 secondes. C'est à la 209ème seconde que la route se reforme et que le trafic TCP est rétabli.

Le graphique de la FIG.4.12 met quant à lui en évidence l'évolution du temporisateur de retransmission de TCP. Il reste généralement constant à sa valeur minimale lorsque le nœucl isolé permet de retransmettre le trafic TCP; les petites montées du temporisateur de retransmission sont uniquement consécutives à la perte isolée de paquets TCP. Ces pertes peuvent aussi être corrélées avec les chutes ponctuelles de débit de la connexion TCP. Lorsque la partition se crée, les paquets ne savent plus être acheminés, ils

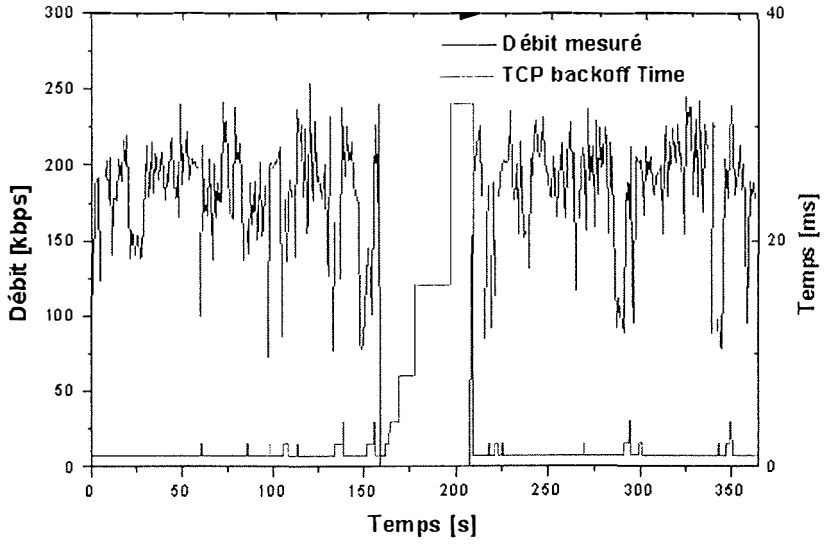


FIG. 4.12 – Evolution du temporisateur de retransmission de TCP .

sont donc jetés par les nœuds intermédiares, ce qui fait augmenter le temporisateur de retransmission. En fait, cette augmentation de temporisateur est la réaction typique de TCP face à la perte de paquets. Le problème de TCP dans cette situation est qu'il assimile la perte de paquets à une indication de congestion. Dans ce cas, il adopte une attitude conservatrice en retardant l'envoi des prochains paquets TCP. La réaction choisie par TCP n'est bien sûr pas la meilleure dans cette situation et montre donc la principale limite de TCP à s'adapter dans un environnement sans fil tel que celui des MANETs.

4.5 Résumé

A l'aide du simulateur ns2, nous avons réalisé un ensemble de manipulations, donnant les résultats suivants : les deux premières séries de manipulations ont montré que le débit est influencé par le nombre de nœuds intermédiaires composant le chemin à travers le réseau pour rejoindre la destination. En effet, lorsque le nombre de sauts augmente, le débit diminue pour se stabiliser à partir de 5 sauts. De plus, la taille des segments a également un impact sur le débit : un segment de grande taille donne un débit plus élevé. On a également remarqué une interaction entre TCP et IEEE 802.11, cette interaction ayant des effets sur le débit.

La troisième série de manipulations observe l'impact de la mobilité sur les performances de TCP. On remarque qu'une augmentation de la vitesse du nœud mobile diminue le débit et augmente le taux de perte. La taille de la fenêtre peut, quant à elle, être mise en rapport avec le taux de perte.

Enfin, la quatrième série de manipulations examine le comportement de TCP face à la création temporaire d'une partition, l'émetteur et le receveur se trouvant dans des partitions distinctes. On remarque que lorsque la partition se crée, TCP réagit comme s'il était en présence de congestion dans le réseau, ajustant son temporisateur de retransmission face aux pertes de paquets consécutives aux défauts de routes engendrés par la partition.

Conclusions

Un réseau mobile ad hoc (MANET) est une collection de nœuds mobiles où il n'existe aucune infrastructure ou administration centralisée. De tels réseaux peuvent se révéler très utiles dans les situations où il n'est pas souhaitable voire impossible de déployer une infrastructure fixe. Dans ce type d'environnement, la cohérence au sein du réseau est assurée à l'aide de protocoles de routage qui se chargent d'acheminer, à travers le réseau ad hoc, les paquets vers leur destination. Plusieurs types de protocoles sont disponibles, chacun possédant ses propres caractéristiques. Cependant, même si ces protocoles fournissent une approche consistante des MANETs au niveau de la couche réseau, il ne fournissent aucune garantie quant à la fiabilité du transport de bout-à-bout. Pour cela, on utilisera TCP, qui est un protocole de transport orienté connexion qui gère l'acheminement des informations entre deux entités terminales de façon fiable.

Ce mémoire est consacré à l'étude du comportement de TCP dans un environnement de type MANET. A l'aide du simulateur ns2, nous avons effectué plusieurs séries de manipulations regroupées en différents scénarii. DSR a été choisi comme protocole de routage ad hoc. Dans chaque simulation, l'entiéreté du trafic est généré par une seule connexion TCP.

A travers tous ces scénarii ainsi envisagés, nous pouvons donner les conclusions suivantes : le débit est fortement influencé par le nombre de sauts à effectuer à travers le réseau. Pour les routes d'une longueur inférieure ou égale à 4 sauts, le débit est réduit de moitié chaque fois que l'on augmente la longueur du chemin d'un intermédiaire. A partir de 5 sauts, on remarque une tendance à la stabilisation du débit, qui continue néanmoins à baisser légèrement. Le débit dépend aussi de la taille des segments. Un segment de plus grande taille donne un débit plus élevé. La taille des segments a également une influence sur le taux de perte : une grande taille de segment donne un taux de perte plus élevé. La taille de la fenêtre ne semble pas avoir d'influence sur le débit, du moins pour les valeurs qui ont été simulées (32 et 64 KBytes). Cependant, une corrélation entre le taux de perte et la taille de la fenêtre a été observée : avec une taille de segment identique, lorsque la taille de la fenêtre augmente le taux de perte augmente aussi. La mobilité

des entités à l'intérieur du réseau joue un rôle à la fois sur le débit et sur le taux de perte : lorsque la vitesse moyenne de déplacemement des nœuds augmente, le débit diminue et le taux de perte augmente proportionnellement. La surcharge générée par l'échange des informations de routage reste raisonnable. Cet élément n'a, en général, qu'une influence ponctuelle sur le débit, principalement lors des recherches de route. Enfin, les performances de TCP sont influencées par la couche MAC. Elle limite le débit en insérant un en-tête et surtout un délai de latence dû à l'accès au canal de transmission. Elle est aussi responsable d'un phénomème d'oscillation de débit consécutif aux tentatives infructueuses d'accès au canal de transmission.

De manière générale, on concluera que le comportement de TCP[2] n'est pas tout à fait adapté à l'environnement MANET. Le principal défaut de TCP est d'interpréter toute perte dans le réseau comme un signal de congestion à l'intérieur de celui-ci. Or, dans le cas, par exemple de perte suite à un défaut de route, cette réaction de la part de TCP sera loin d'être optimale. Il faudrait donc adapter le comportement de TCP en tenant des événements spécifiques aux environnements MANET. A cette fin, signalons que des implémentations de TCP «adaptées» pour de tels environnements sont déjà disponibles, [15, 21, 23] en sont quelques exemples.

Enfin, voici quelques propositions en vue de poursuivre le travail qui a été effectué : d'abord, pour l'ensemble des simulations effectuées, un seul protocole de routage a été utilisé. Vu la grande diversité des protocoles de routage proposés, il serait intéréssant de comparer les performances de TCP avec plusieurs de ces protocoles de routage. Ensuite, les scénarii envisagés étaient volontairement simplifiés afin de pouvoir valider les résultats obtenus. Maintenant que les résultats de base sont acquis, il pourrait être envisageable de complexifier un peu les scénarii afin de se placer dans des conditions un peu plus réalistes, en proposant : le remplacement du trafic FTP précédemment utilisé par du trafic WEB, cohabitation avec un réseau fixe via des points d'accès ou encore simulation sur des topographies plus «réalistes» (prise en compte de phénomène d'écrantage, ...). Enfin, il serait intéréssant de tester des implémentations de TCP qui ont été adaptées aux environnements MANET. Malheureusement, à l'heure actuelle, aucun simulateur ne propose de telles implémentations.

[2]Il est question ici de la version SACK, implémentation utilisée lors de nos simulations.

Annexe A
Glossaire

CSMA/CA : Carrier Sense Multiple Access with Collision Avoidance
CWND : Congestion Window
CTS : Clear To Send
DARPA : Defense Advanced Research Project
DSR : Dynamic Source Routing
DSSS : Direct-Sequence Spread Spectrum
FTP : File Transfert Protocol
GPRS : General Packet Radio Service
IEEE : Institute of Electrical and Electronic Engeneering
IETF : Internet Engeneering Task Force
LPR : Low-cost Packet Radio
MAC : Medium Access Control
MANET : Mobile Ad hoc NETwork
PAN : Personnal Area Network
PDA : Personnal Digital Assistant
PRNet : Packet Radio Network
RERR : Route ERRor
RREP : Route REPly
RREQ : Route REQuest
RTS : Request To Send
RTT : Round Trip Time
SURAN : SUrvivable RAdio Network
TCP : Transmission Control Protocol
UMTS : Universal Mobile Telecommunication System
WLAN : Wireless LAN

Annexe B
Ns2

Le network simulator (ns2)[19] est un simulateur de réseau à événement discret développé par l'Université de Berkeley en Californie et le projet VINT. Ce simulateur fournit un support pour l'étude de TCP et du routage à travers des réseaux cablés et sans fil. Le groupe de recherche Monarch de l'Université Carnegie-Mellon (CMU) a développé une extension au simulateur pour les MANETs, qui modélise les couches physique, liaison de données et MAC. L'ensemble comprenant le simulateur avec l'extension peut être téléchargé par Internet [24].

Ns2 est implémenté à l'aide de deux langages de programmation : le noyau du simulateur est implémenté en C++, alors que l'interaction avec le simulateur (instanciation des objets, paramétrages des variables) s'effectue à l'aide de scripts TCL. Cette approche tente à favoriser la productivité de l'utilisateur en utilisant un langage simple pour configurer le simulateur tout en possédant un noyau implémenté dans un langage efficace pour effectuer les simulations.

Les principales caractéristiques de ns2 sont :
- le support de protocoles ad hoc. Quatres protocoles sont disponibles : DSDV, DSR, AODV et TORA ;
- un format de trace détaillé et adapté aux nœuds mobiles mais aussi des outils de post-simulation tels que NAM pour visualiser les traces ou xgraph pour tracer des variables du simulateur ;
- ns2 est extensible. Il est possible de le personnaliser et/ou d'y ajouter de nouvelles fonctionnalités (nouveaux protocoles, ...).

Les principaux inconvénients de ns2 sont le manque de complétude de la documentation et son apprentissage difficile et donc très lent.

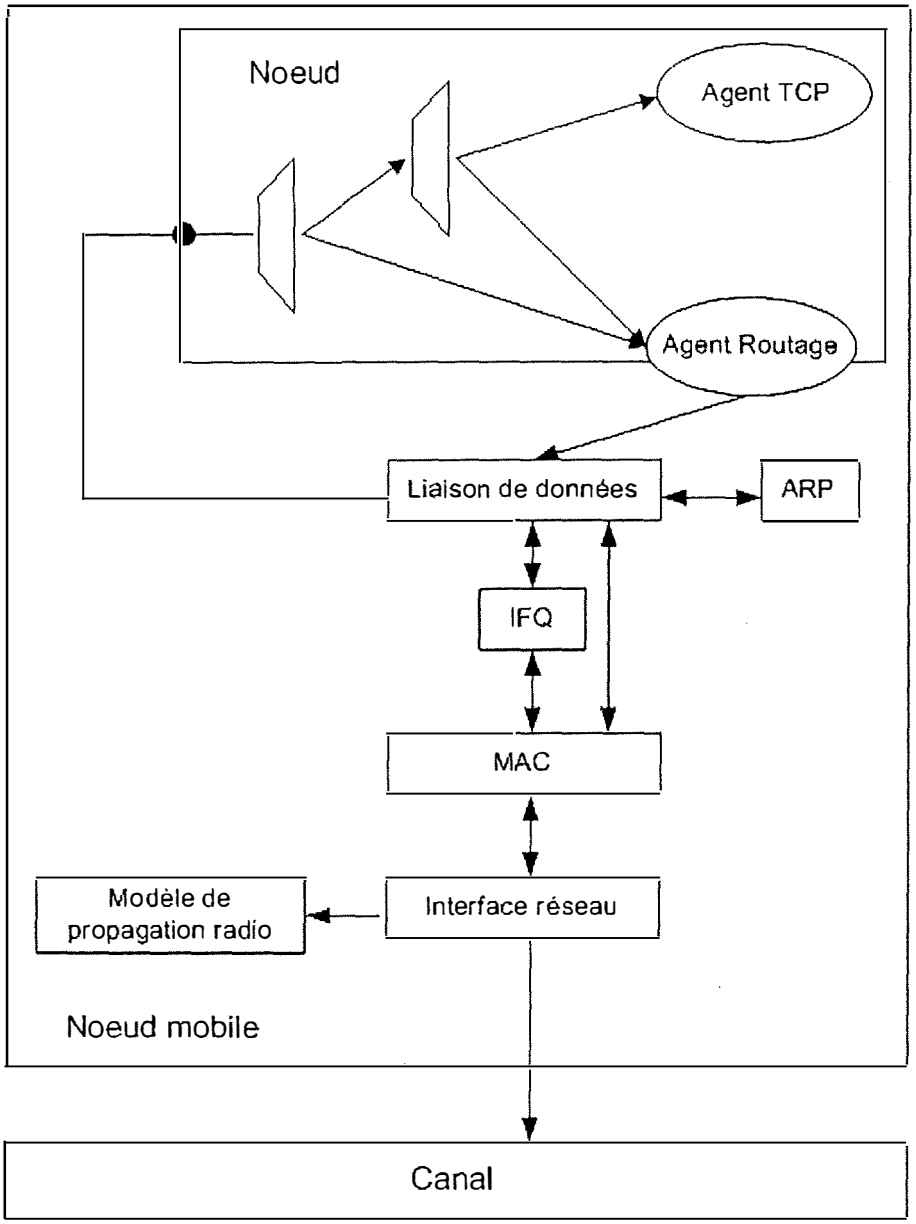


FIG. 13 – Schéma de principe d'un nœud mobile dans ns2.

Composants dans les réseaux ad hoc

Dans ns2, chaque entité d'un réseau ad hoc est appelé «nœud mobile». La pile contenue dans chaque nœud mobile consiste en une couche liaison de données (LL), une queue pour l'interface radio (IFQ), une couche MAC et une interface réseau (netIF).

Regardons maintenant un peu plus en détails ces éléments :

couche liason de données : cette couche possède un module ARP dont le rôle est de convertir les adresses IP en adresses hardware (MAC). Normalement, tous les paquets sortant sont dirigés vers la couche LL par l'agent de routage. La classe LL est implémentée dans */ns-allione-2.1b8/ll.cc,h.*

ARP : le module de Résolution d'Adresse traite les requêtes provenant de la couche liaison de données. Si le module ARP possède l'adresse hardware de la destination, il l'écrit dans l'en-tête MAC du paquet. Sinon, une ARP Query est envoyée en broadcast et le paquet est temporairement stocké dans un buffer. Pour chaque destination inconnue, il existe un buffer d'une taille d'un paquet. Si plusieurs paquets doivent être stockés dans le buffer, c'est le paquet le plus ancien qui sera jeté. Dès que l'adresse du prochain saut est connue, le paquet est inséré dans la queue de l'interface radio (IFQ). Le module ARP est implémenté dans */ns-allione-2.1b8/arp.cc,h*

Interface Queue : la classe Priqueue est implémentée comme une queue à priorité qui donne la priorité aux paquets de routage, en les insérant en tête de la queue. Elle possède également un filtre sur les paquets présents dans la queue qui permet de retirer les paquets possédant une certaine adresse. La classe priqueue est implémentée dans */ns-allione-2.1b8/priqueue.cc,h.*

Couche MAC : l'implémentation de la couche MAC utilise les mécanismes de physical et virtual carrier sense. La couche MAC utilise donc le pattern RTS/CTS/DATA/ACK pour les paquets unicast et envoie des paquets DATA pour les messages broadcast. L'implémentation de la couche MAC est disponible dans */ns-allione-2.1b8/mac-802-11.cc,h.*

Interface réseau : la couche interface réseau sert d'interface hardware vers le canal de transmission. Cette interface est sujette aux collisions en recevant les paquets reçus par les autres nœuds mobiles. L'interface marque chaque paquet transmis avec des méta-données relatives à l'interface de transmission telles que la puissance de transmission et la longueur d'onde. Ces méta-données se trouvent dans l'en-tête des paquets transmis et sont utilisées par le modèle de propagation pour savoir si l'onde électromagnétique associée au paquet possède la puissance nécessaire pour être captée par le nœud d'arrivée. Le modèle approxime le fonctionnement d'une interface radio de type Direct-Sequence Spread Spectrum (DSSS) de Lucent WaveLen.

Cette classe est implémentée dans */ns-allione-2.1b8/wireless-phy.cc,h.*

Modèle de propagation radio : le modèle de propagation radio utilise l'équation de Friss pour une atténuation en $1/r^2$ à courte distance et un modèle de Two Rays Ground Reflection en $1/r^4$ à plus grande distance (voir annexe B). Cette classe est implémentée dans */ns-allione-2.1b8/tworayground.cc,h.*

Antenne : une antenne omni-directionnelle de gain unitaire est utilisée dans chaque nœud mobile. Cette antenne est centrée sur le nœud mobile et placée à une hauteur de 1,5 mètre. Cette classe est implémentée dans */ns-allione-2.1b8/antenna.cc,h.*

Canal : le canal duplique les paquets vers tous les nœuds qui lui sont attachés à l'exception de l'émetteur du paquet. C'est le rôle du receveur de détecter le paquet ainsi transmis.

Annexe C
Méthode expérimentale

Le but de cette annexe est de donner quelques détails supplémentaires sur le déroulement des simulations sous ns2.

Déroulement d'une simulation sous ns2

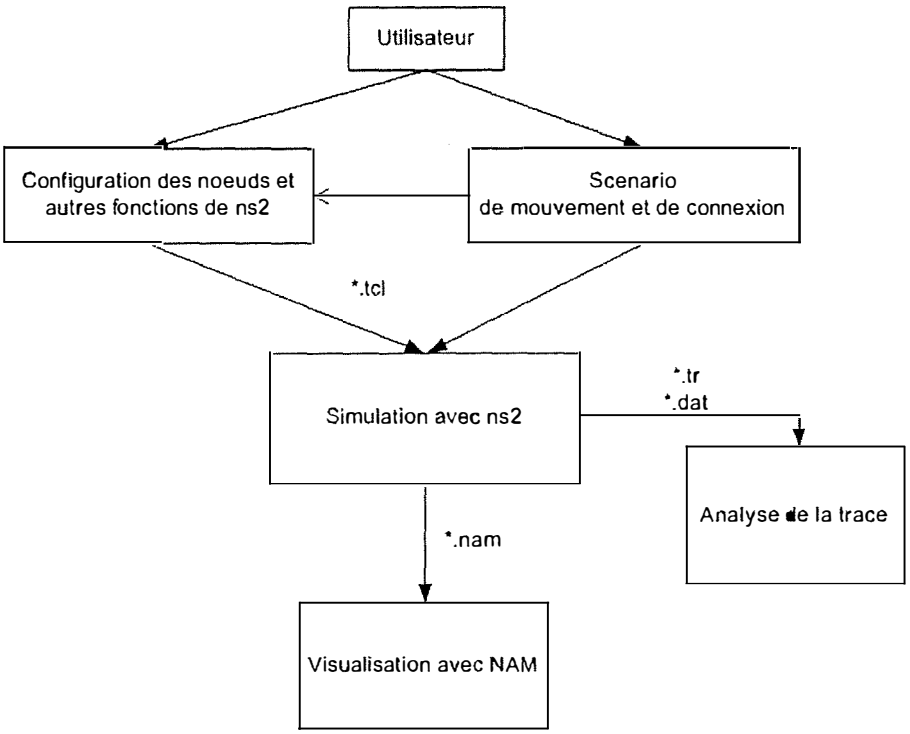


FIG. 14 – Déroulement d'une simulation avec ns2.

Le schéma de la FIG.14 reprend la démarche à suivre pour effectuer une simulation de réseau ad hoc avec ns2. Il faut tout d'abord commencer par créer un fichier *.tcl dans lequel on reporte les paramètres généraux de la

simulation, c'est-à-dire la configuration complète des noeuds mais aussi la gestion des principaux événements du simulateur (démarrage/arrêt de la simulation, démarrage de la procédure de traçage des variables, initialisation des fichiers d'entrée et de sortie). A côté de cela, il est important de créer deux fichiers «scénario» : le premier est le scénario de mouvement, qui correspond au mouvement qu'on souhaite donner aux différents nœuds du réseau. Le second est le scénario de connexion dans lequel on initialise tous les paramètres de la connexion (agent de transport, génération du trafic, procédure de traçage des variables).

Lorsque ces fichiers sont créés, on lance une instance du simulateur en passant comme fichier d'entrée le nom du fichier *.tcl que l'on vient de créer. Le résultat de cette opération est un ensemble de fichiers de sortie :

- un fichier *.tr qui représente la trace générée par le simulateur. Ce fichier trace le comportement de chaque paquet de données échangé durant la simulation ;
- un fichier *.nam qui est destiné à l'utilitaire nam, une visionneuse permettant de suivre en direct l'évolution du réseau durant la période de simulation ;
- un fichier *.dat qui reprend l'évolution des variables qui ne sont pas reprises dans la trace mais que l'on souhaite tout de même analyser.

Les deux sections suivantes présentent les principaux fichiers d'entrée et de sortie.

Fichiers d'entrée

Fichier «sim.tcl» :

```
# sim.tcl:
#
# ======================================================================
# Options de simulation
# ======================================================================

#Configuration générale
set val(chan)       Channel/WirelessChannel
set val(prop)       Propagation/TwoRayGround
set val(netif)      Phy/WirelessPhy
set val(mac)        Mac/802_11
set val(ifq)        Queue/DropTail/PriQueue
set val(ll)         LL
set val(ant)        Antenna/OmniAntenna
```

```
set val(x)              2000          ;# Topographie: Xmax
set val(y)              2000          ;# Topographie: Ymax
set val(nsi)            20            ;
set val(ifqlen)         50            ;# Nombre max de packet dans IFQ
set val(seed)           0.0
set val(adhocRouting)   DSR           ;# Protocole de routage
set val(nn)             10            ;# Nombre de noeuds dans la sim
set val(cp)             "./scenario/cp-manip2a"   ;# Scenario de connexion
set val(sc)             "./scenario/mp-manip2-12" ;# Scenario de mouvement
set val(tr)         "manip1.tr"   ;# Fichier trace
set val(nm)         "manip1.nam"  ;# Fichier nam
set val(out)            "manip2-out.dat"     ;# Fichier variables
set recinterval         0.01          ;# Intervalle enregistrement données
set val(stop)            1000.0       ;# Temps max de simulation


# Configuration couche liaison de données
LL set mindelay_ 50us
LL set delay_ 25us


# Configuration antenne
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0       ;# Gain antenne (transmission)
Antenna/OmniAntenna set Gr_ 1.0       ;# Gain antenne (réception)

# Configuration de l'interface physique, configurée pour simuler
# un équipement de type Lucent WaveLAN DSSS 914MHz
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10   ;# Seuil de réception
Phy/WirelessPhy set Rb_ 2*1e6
Phy/WirelessPhy set Pt_ 0.2818            ;# Puissance de l'émetteur
Phy/WirelessPhy set freq_ 914e+6          ;# Fréquence
Phy/WirelessPhy set L_ 1.0                ;# Perte du système

# ======================================================================
# Programme principal
# ======================================================================

# Nouvelle instance du simulateur
set ns_ [new Simulator]
```

```
# Nouvelle topographie
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Initialisation des fichiers de sortie
set tracefd [open $val(tr) w]
set namtrace    [open $val(nm) w]
set outfile     [open $val(out) w]

$ns_ use-newtrace          ;# Utilisation du nouveau format de trace
$ns_ trace-all $tracefd    ;# Tous les événements sont tracés
#$ns_ namtrace-all-wireless $namtrace $val(x) $val(y) ;# Traçage pour nam

# Création d'un GOD (General Operation Director)
set god_ [create-god $val(nn)]

# Nouvelle api pour l'initialisation du canal
# Création du canal #1 (tous les noeuds sont sur le même canal)
set chan_1_ [new $val(chan)];



#=============================================================================
# Configuration des noeuds mobiles
#=============================================================================

$ns_ node-config -adhocRouting $val(adhocRouting) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -topoInstance $topo \
                 -channel $chan_1_\
                 -movementTrace OFF \
                 -agentTrace ON \
                 -routerTrace OFF \
                 -macTrace OFF

# Création du nombre souhaité de noeuds.
for {set i 0} {$i < $val(nn) } {incr i} {
```

```
set node_($i) [$ns_ node]
$node_($i) random-motion 0 ;# Mouvement aléatoire désactivé
}


#==========================================================================
# Chargement des scenarii de connexion et de mouvement
#==========================================================================

puts "Chargement scenario de connexion ($val(cp))..."
source $val(cp)

puts "Chargement scenario de mouvement ($val(sc))..."
source $val(sc)



#==========================================================================
# Evenements particuliers du simulateur
#==========================================================================

# Ecriture de l'en-tête du fichier trace
puts $tracefd "M 0.0 nn $val(nn) x $val(x) y $val(y) rp $val(adhocRouting)"
puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed $val(seed)"
puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"


# Configuration de la taille des noeuds dans nam
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) $val(nsi)
}

# Avertissement lancé aux noeuds quand la simulation se termine
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i) reset";
}

# Démarrage de la simulation
puts "Starting Simulation..."
$ns_ run

# Arrêt de la simulation
$ns_ at  $val(stop).0002 "puts \"NS EXITING...\" ;  $ns_ halt"
```

Fichier «scénario de connexion» :

```
# Initialisation des agents TCP (émetteur et receveur)
set tcp [new Agent/TCP/Sack1]
set sink [new Agent/TCPSink/Sack1]


$tcp set packetSize_ 1460  ;# Taille des segments TCP
$tcp set window_ 44  ;# Taille de la fenêtre (en nombre de packets)
$tcp set maxcwnd_ 44 ; #Taille max de la fenêtre de congestion

# Attachement des agents TCP aux noeuds mobiles désignés
$ns_ attach-agent $node_(0) $tcp
$ns_ attach-agent $node_(1) $sink


# Création du trafic
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ connect $tcp $sink
$ftp send -1 ;# Transfert infini = les buffers de l'émetteur
              # sont toujours remplis

#================================================================
# Procédure de traçage de certaines variables d'état de TCP
#================================================================

proc record {} {
    global ns_ outfile recinterval tcp
    set now [$ns_ now]
    puts $outfile "$now  [$tcp set rtt_] [$tcp set cwnd_]
                   [$tcp set backoff_]"
    $ns_ at [expr $now+$recinterval] "record"
}



# On commence à générer du trafic dès le début de la simulation
$ns_ at 0.0 "$ftp start"
```

```
# On enregistre de l'information sur les variables d'états sélectionnées
# dès le début de la simulation
$ns_ at 0.0 "record"
```

Fichier «scénario de mouvement» :
Ce fichier est composé de deux parties :
- une partie de configuration où on donne la position initiale de chacun des nœuds participant à le simulation ;
- une partie où on décrit le mouvement des nœuds mobiles.

Les lignes ci-dessous donnent un exemple d'un tel fichier.

```
# Position initiale de chaque noeud
$node_(0) set Z_ 0
$node_(0) set Y_ 1200
$node_(0) set X_ 1000
        ...
$node_(18) set Z_ 0
$node_(18) set Y_ 1500
$node_(18) set X_ 1300
        ...
# mouvement de chaque noeud pendant la simulation
$ns_ at 292.5 "$node_(5) setdest 2560  2285 5000.0"
$ns_ at 341 "$node_(0) setdest 1682  1200 5000.0"
```

Fichier de sortie

Fichier «trace» :
Le fichier trace (*.tr) reporte tous les événements survenus durant la simulation au niveau de l'échange des paquets, à trois niveaux différents : la couche MAC, l'agent de routage, l'agent de transport.

Le format de trace utilisé est un nouveau format spécialement développé pour les nœuds mobiles. La FIG.15 en donne la spécification. Dans ce type de trace, la dernière partie de la trace est spécialisée suivant le type de paquet dont il s'agit (TCP, UDP, DSR, TORA, ...). Afin de garder la figure lisible, seul le cas des paquets TCP a été indiqué.

Voici maintenant un exemple de trace prise au niveau de l'agent de transport.

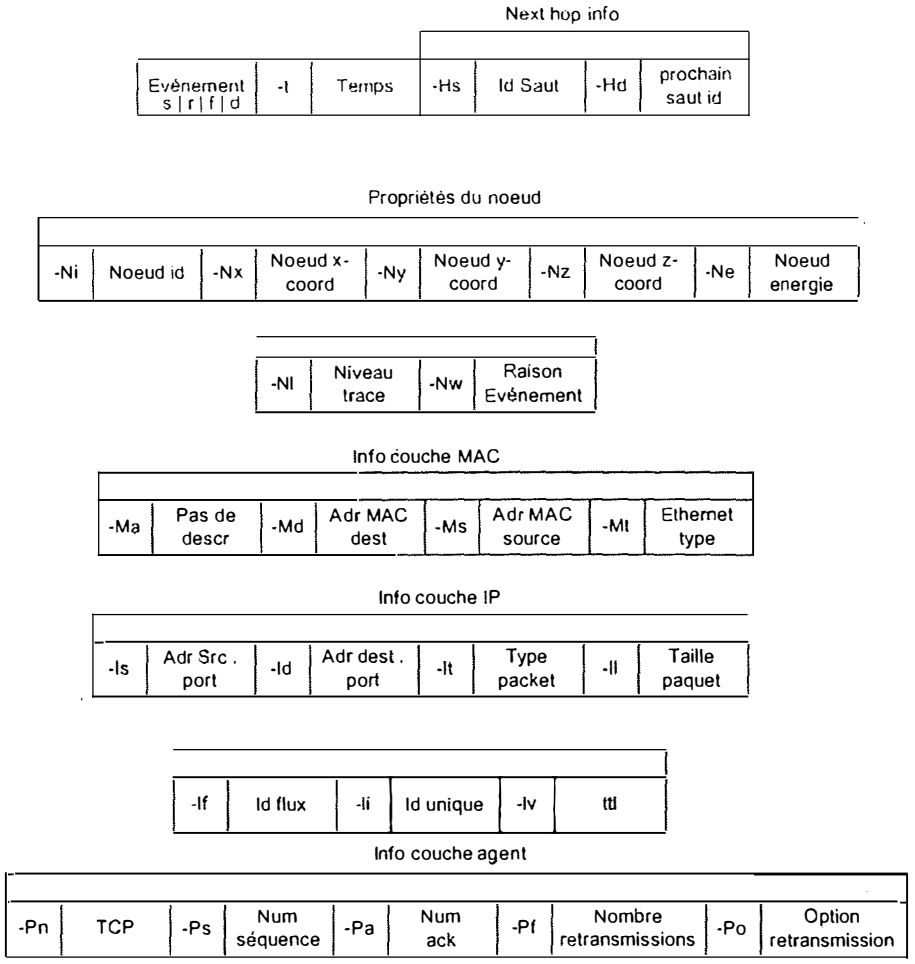


FIG. 15 – Spécification d'une trace pour les nœuds mobiles. Cas d'un segment TCP.

```
s -t 361.810251114 -Hs 1 -Hd -2 -Ni 1 -Nx 1000.00 -Ny 1000.00
-Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0
-Is 1.0 -Id 0.0 -It ack -Il 40 -If 0 -Ii 50900 -Iv 32 -Pn tcp
-Ps 18846 -Pa 0 -Pf 0 -Po 0
r -t 361.821286047 -Hs 1 -Hd 1 -Ni 1 -Nx 1000.00 -Ny 1000.00
-Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma a2 -Md 1 -Ms 2 -Mt 800
-Is 0.0 -Id 1.0 -It tcp -Il 1508 -If 0 -Ii 50892 -Iv 32 -Pn tcp
-Ps 18847 -Pa 0 -Pf 6 -Po 0
s -t 361.821286047 -Hs 1 -Hd -2 -Ni 1 -Nx 1000.00 -Ny 1000.00
-Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0
-Is 1.0 -Id 0.0 -It ack -Il 40 -If 0 -Ii 50901 -Iv 32 -Pn tcp
```

```
-Ps 18847 -Pa 0 -Pf 0 -Po 0
r -t 361.828147781 -Hs 1 -Hd 1 -Ni 1 -Nx 1000.00 -Ny 1000.00
-Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma a2 -Md 1 -Ms 2 -Mt 800
-Is 0.0 -Id 1.0 -It tcp -Il 1508 -If 0 -Ii 50894 -Iv 32 -Pn tcp
-Ps 18848 -Pa 0 -Pf 6 -Po 0
```

Les deux autres fichiers de sortie ne seront pas illustrés dans ce mémoire. Le fichier destiné à l'utilitaire nam est simplement un fichier trace qui a été formaté pour être reconnu par ce dernier. Le fichier *.dat reprend simplement un ensemble de colonnes décrivant l'évolution des variables que l'on veut suivre. Ce fichier est principalement destiné à être visualiser à l'aide d'un traceur tel que Gnuplot par exemple.

Bibliographie

[1] H. Abrahamsson. Traffic measurement and analysis. SICS Technical Report T99/05, 1999.

[2] Mobile Ad-Hoc Networks (MANET) charter. http ://www.ietf.org/html.charters/manet-charter.html.

[3] M. Mathis J. Mahdavi S. Floyd et A.Romanow. Tcp selective acknowledgement options. RFC2018, Avril 1996.

[4] P. Karn et C. Partridge. Improving round-trip estimates in reliable transport networks. Proceedings of SIGCOMM 1987, ACM, 1987.

[5] V. Jacobson R. Braden et D. Borman. Tcp extensions for high performance. RFC1323, Mai, 1992.

[6] D. Johnson et D. Maltz. The dynamic source routing in ad hoc wireless networks.

[7] J. Brosh D. Johnson et D. Maltz. The dynamic source routing protocol for mobile ad hoc networks. draft-ietf-manet-dsr-*.txt.

[8] S. Corson et J. Macker. Mobile ad hoc networking (manet) : Routing protocol performance issues and evaluations considérations. RFC 2510.

[9] G. Holland et N. Vaidya. Analysis of tcp performance over mobile ad hoc networks. Proceeding of Mobicom'99, Seattle.

[10] M. Frodish P. Johansson et P. Larson. Wireless ad hoc networking : the art of networking without a network. Ericsson Review, no 4, 2000.

[11] M. Gerla K. Tang et R. Bagrodia. Tcp performance in wireless multi-hop networks. Proceeding of IEEE WMSCA'99, New Orleans.

[12] J. Li C. Blake H. Lee et R. Morris. Capacity of ad hoc networks.

[13] K. Chandran S. Raghunathan S. Venkatesan et R. Prakash. A feedback based scheme for improving tcp performance in ad-hoc wireless networks. International Conference on Distributed Computing Systems, 1997.

[14] S. Xu et T.Saadawi. Does the ieee 802.11 mac protocol work well in multihop wireless ad hoc networks ? IEEE Communication Magazine, Juin, 2001.

[15] D. Kim C.K. Toh et Y. Choi. Tcp-bus : Improving tcp performance in wireless ad hoc networks. Journal of Communications and Networks, Vol. 3, no 2, Juin, 2001.

[16] A. Fladenmuller. Les réseaux ad hoc. Tutorial, Laboratoire LIP6.

[17] L. Freeney. A taxonomy for routing protocols in mobile ad hoc networks. SICS Tech Report T99/07, Octobre, 1999.

[18] V. Jacobson. Congestion avoidance and control. Proceedings of SIGCOMM 1988, ACM, 1988.

[19] editeurs K. Fall et K. Vanahan. Ns notes and documentation. The VINT Project, UC Berkeley, LBL, USC/ISI et Xerox PARC, Janvier 2002.

[20] T. Lemlouma. Le routage dans les réseaux mobiles ad hoc. Master Thesis, Septembre, 2000.

[21] J. Liu. Atcp : Tcp for mobile ad hoc networks. IEEE Journal on selected areas in communications, Vol. 19, No. 7, Juillet, 2001.

[22] S. Medidi. Mobile ad hoc networks - manets. 2002.

[23] S. Narayanaswamy. Tcp for wireless networks. Independent Study Report, Mai, 2001.

[24] ns2 nanual. http ://www.isi.edu/nsnam/ns/doc/index.html.

[25] C. Perkins. *Ad Hoc Networking.* Addison-Wesley, 2001.

[26] J. Postel. Transmission control protocol darpa internet program protocol specification. RFC793, Septembre, 1981.

[27] L. Qin. pro-active route maintenace in dsr. Master Thesis, Août, 2001.

[28] B. Shrader. A proposed definition of 'ad hoc network'. Mai, 2002.

[29] W. Stevens. Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. RFC2001, Janvier, 1997.

[30] W. Stevens. *TCP/IP Illustrated, Volume 1 : The Protocols.* Addison-Wesley, 1996.

[31] A. Tanenbaum. *Computer Networks.* Prentice-hall International, 3rd edition, 1996.

[32] N. Vaidya. Mobile ad hoc networks : Routing, mac and transport issues. Tutorial, 2001, http ://www.cs.tamu.edu/faculty/vaidya/.